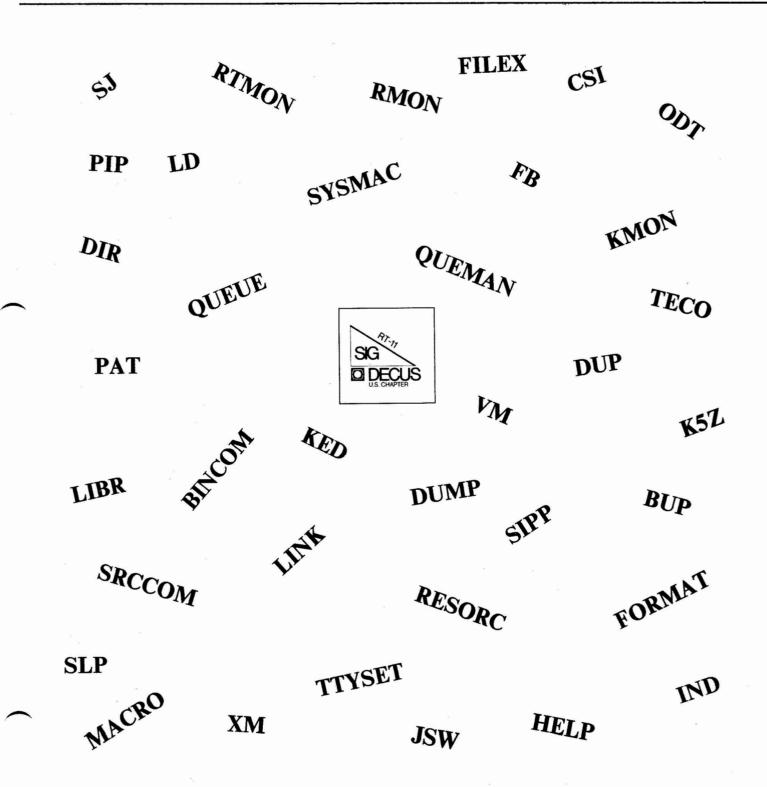
# minita/ker

DECUS

October 1983

Volume 9, Number 4



Printed in the U.S.A.

The following are trad	emarks of Digital Equipment (	Corporation:
DEC	DIBOL	PDT
DECnet	Digital Logo	RSTS
DECsystem-10	EduSystem	RSX
DECSYSTEM-20	IAS	UNIBUS
DECUS	MASSBUS	VAX
DECwriter	PDP	VMS
		VT

UNIX is a trademark of Western Electric Corporation

Copyright© Digital Equipment Corporation 1983 All Rights Reserved

It is assumed that all articles submitted to the editor of this newsletter are with the authors' permission to publish in any DECUS publication. The articles are the responsibility of the authors and, therefore, DECUS, Digital Equipment Corporation, and the editor assume no responsibility or liability for articles or information appearing in the document. The views herein expressed are those of the authors and do not necessarily express the views of DECUS or Digital Equipment Corporation.

# THE DECUS DECUS IN RT-11 SIG NEWSLETTER

October 1983

Volume 9, Number 4

### TABLE OF CONTENTS

-----

ADDRESSES	
Newsletter Submissions	3
FROM THE EDITOR	
Volunteers Needed	4
USER INPUT	
Documentation Directory For RT-11 Device Handlers	4
Patch To Allow RT-11 V5.0 RX03 Support	11
Bit Manipulation Program	12
RT-11 V5.0 Extended Memory Patch	19
Cursor Driven Command File Driver	23
USER REQUESTS	
FRT.MAC Question	.34
RT-11 V3 Serial Printer Handler	34
USER RESPONSES	
Change to DATE Program	34
UPCOMING SYMPOSIUM INFORMATION	
RT-11 Session Notes	35
Las Vegas Schedule	36
PAST SYMPOSIUM INFORMATION	
306A Clock Board Support	38
RT-11 Macro/Fortran Interactions	46
MACDBG/RT-11: A User's Critique	48
Creation and Handling of Multi-Volume Directories	49
SOFTWARE PERFORMANCE REPORTS	
SIPP/PIP/Backup Problem	50
SYMPOSIUM TAPE INFORMATION	
Call For SIG Tape Submissions	51
Tape Copy Release Agreement	51
Tape Copy Release Form	52



DECUS

October 1983

Volume 9, Number 4

Contributions to the newsletter should be sent to:

Ken Demers MS-48 United Technologies Research Center East Hartford, Ct. 06108 (203) 727-7139 or 7240

Other communications can be sent to:

John T. Rasted JTR Associates 58 Rasted Lane Meriden, Ct. 06450 (203) 634-1632

RT-11 SIG C/O DECUS One Iron Way MR2-3/E55 Marlboro, Ma. 01752 (617) 467-4141

#### From The Editor

I still need more volunteers to convert the audio tapes recorded at DECUS Symposium RT-11 sessions into articles for the "Minitasker". You will only be responsible for converting the tape from any one session. Please contact me as soon as possible.

Thankyou.

Ken Demers

#### USER INPUT

- DSIR \_\_\_\_\_\_ DEPARTMENT OF SCIENTIFIC AND INDUSTRIAL RESEARCH

#### **APPLIED MATHEMATICS DIVISION**

P.O. Box 1335 Wellington New Zealand Telephone (4) 727 855 Telex 3276 Research 7th Floor Rankine Brown Building Victoria University of Wellington 

SCIN 123:1

3 MAY 1983

A DOCUMENTATION DIRECTORY FOR RT-11 DEVICE HANDLERS AND INTERRUPTS. 

#### R. D. BROWNRIGG

ABSTRACT: An exhaustive list is presented of those references to device handlers and interrupt processing contained in the DEC documentation available for the RT-11 version 4.0 operating system.

1. INTRODUCTION.

The following references are to section numbers, figures, and tables contained in the various manuals available for the RT-11 version 4.0operating system. All references have direct relevance to device handlers in particular or interrupt processing in general. In some cases, further information is provided in parentheses to clarify exactly which aspects of these topics are mentioned or discussed in the particular section referred to. Page numbers are also provided.

The manuals referred to and their abbreviations are as follows: - DIGITAL 'microcomputers and memories' handbook (1982 edition) MAM - RT-11 Installation and System Generation Guide GEN

- SYS - RT-11 System User's Guide
- REF
- RT-11 Programmer's Reference Manual
- SUP - RT-11 Software Support Manual

4

2. DIGITAL 'microcomputers and memories' handbook

ARCHITECTURAL OVERVIEW PROCESSOR STATUS WORD (interrupt priority)		CHAPTER CHAPTER	_
PROGRAMMING TECHNIQUES POSITION INDEPENDENT CODE (virtual address space) STACKS (subroutines, interrupts) INTERRUPTS (interrupt enable bit)	MAM MAM	CHAPTER CHAPTER CHAPTER CHAPTER	8 8
LSI-11 BUS INTERRUPTS (vector, device priority)		CHAPTER CHAPTER	-
MEMORY MANAGEMENT (kernel mode, user mode) MEMORY RELOCATION PROTECTION PAGE ADDRESS REGISTER (PAR) VIRTUAL AND PHYSICAL ADDRESSES	MAM MAM MAM	CHAPTER CHAPTER CHAPTER CHAPTER CHAPTER	10 10 10

3. RT-11 Installation and System Generation Guide

1.1.3.2 Do You Need to Perform the System Generation Process? (device I/O timeout, error logging, extra device slots) GEN 1-14 Table 1-5 Features Available Only Through System Generation Process (device I/O timeout, error logging) GEN 1-15 2.8.13 Installing Other Devices (RT-11 bootstrap action) GEN 2-31 8.2.3 Monitor Services for Target Applications (device timeout, error logging) GEN 8-5 8.3 Studying the SYSGEN dialogue (device I/O timeout, error logging, extra device slots) GEN 8-15 F.1 System Conditionals (device I/O timeout, error logging) GEN F-1 4. RT-11 System User's Guide SYS 1-5 1.2.2 Device Handlers (definition) 3.3 Physical Device Names (standard names) SYS 3-3 SYS Table 3-1 Permanent Device Names 3-3 SYS 3-5 3.5 Device Structures (random access, sequential access) 4.4 SYS 4-15 Keyboard Monitor Commands (INSTALL device handler) SYS 4-112 SYS 4-126 (LOAD device handler) SYS 4-139 (REMOVE device handler) (SET handler characteristics) SYS 4-149 SYS 4-150 Table 4-13 SET Device Conditions and Modification (SHOW device assignments, handler status) SYS 4-160 (UNLOAD device handler) SYS 4-176

17.1 Calling and Using RESORC Table 17-1 RESORC Options 17.2.3 Device Handler Status Option (/D)	SYS SYS	17-1 17-2 17-3
17.2.6 Device Assignments Option (/L)	SYS	17-5
19.1 Uses (error logging) 19.2 Error Logging Subsystem Figure 19-1 Error Logging Subsystem	SYS	19-1 19-2 19-3
5. RT-11 Programmer's Reference Manual		
1.1.2.7 Programmed Request Errors		
(processor status, error byte)	REF	1-12
1.1.3.1 Initialization and Control		1-16
(I/O requests, timer requests, queue elements)	REF	1-15
1.1.3.5 Input/Output Operations (I/O requests, completion routines)	REF	1-19
1.1.3.7 Timer Support (timer requests)	REF	1-23
1.1.3.11 Interrupt Service Routines (.INTEN, .SYNCH)	REF	1-25
1.1.3.12 Device Handlers (special macros)	REF	1-26
2.12 .CTIMIO (cancel device timeout)		
Table 2-1Timer Block Format (I/O timeout)		-
2.16 .DRAST (driver asynchronous trap)		•
2.17 .DRBEG (driver begin)		-
2.18 .DRBOT (driver bootstrap)		2-31
2.19 .DRDEF (driver definitions)		
2.20 .DREND (driver end)		2-33
2.21 .DRFIN (driver finish)		2-34
2.22 .DRSET (driver SET options)		2-34
2.23 .DRVTB (driver vector table)		2-35
2.24 .DSTATUS (device status)		2-36
2.29 .FETCH/.RELEASE (device handler load/unload)		
2.30 .FORK (dismiss interrupt)		2-46
2.37 .HRESET (hardware reset)		2 <b>-</b> 56 2 <b>-</b> 57
2.38 .INTEN (interrupt notify) 2.57 .QELDF (queue element define)	REF REF	
2.57 .QSET (queue element define) 2.58 .QSET (set queue length)		2-87
2.50 .READ/.READC/.READW (read/with completion/with wait)		2-94
2.76 .SPFUN (special function I/O)	REF	2-121
2.79 .SYNCH (synchronise with user state)	REF	2-127
2.80 .TIMIO (I/O timeout)	REF	
2.92 .WRITE/.WRITC/.WRITW (write/with completion/with wait)	REF	2-142
_		
6. RT-11 Software Support Manual		
	OUD	2 8
2.1.3 Interrupt Vectors (standard) Figure 2-4 Interrupt Vector Area	SUP SUP	2 <b>-</b> 8 2 <b>-</b> 10
	001	- IU

Figure 2-4 Interrupt Vector Area	SUP	2-10
2.1.4 I/O Page (addresses)	SUP	2-10
Figure 2-5 I/O Page	SUP	2-11
2.1.5 System Device Handler (bootstrap)	SUP	2-11
Figure 2-6 System Device Handler	SUP	2-12
2.2.1 Device Handlers and Free Space (loading into memory)	SUP	2-19
Figure 2-11 SJ System with Two Loaded Handlers	SUP	2-20
Figure 2-12 SJ System with One Handler Unloaded	SUP	2-21
Figure 2-13 SJ System with Both Handlers Unloaded	SUP	2-22
2.3.4 Size of Device Handlers (where specified)	SUP	2-39

3.2 Clock Support and Timer Service	SUP	3-9
3.2.1 SJ Systems Without Timer Service	SUP	3-9
3.2.2 Systems With Timer Service (timer implementation)	SUP	3-9
Figure 3-5 Timer Queue Element Format	SUP	3-10
3.3 Queued I/O System (components)	SUP	3-11
3.3.1 I/O Queue (structure)	SUP	3-12
Figure 3-6 Components of the Queued I/O System	SUP	3-12
Figure 3-7 I/O Queue Element Format	SUP	3-13
Figure 3-8 I/O Queue with Three Available Elements	SUP	3-14
Figure 3-9 I/O Queue with Two Available Elements	SUP	3-15
Figure 3-10 I/O Queue with One Available Element	SUP	3-15
		_
Figure 3-11 I/O Queue When One Element is Returned		3-16
Figure 3-12 I/O Queue When Two Elements are Returned	SUP	3-16
3.3.2 Completion Queue (structure)	SUP	3-17
Figure 3-13 Device Handler Queue when a New Element is		5
	ann	0 45
Added		3-17
Figure 3-14 Completion Queue Element Format	SUP	3-18
3.3.2.1 SJ Considerations (interruptibility)	SUP	3-18
3.3.2.2 .SYNCH Considerations (format)		3-18
Figure 3-15 Synch Queue Element Format	SUP	3-19
3.3.3 Flow of Events in I/O Processing	SUP	3-19
3.3.3.1 Issuing the Request (blocking)		3-19
	SUP	
3.3.3.2 Queuing the Request in SJ (I/O initiation)	SUP	3-20
3.3.3.3 Queuing the Request in FB and XM		
(system state, holding)	SUP	3-20
3.3.3.4 Performing the I/O Transfer	SUP	3-22
3.3.3.5 Completing the I/O Request	SUP	-
Figure 3-16 Device Handler/Resident Monitor Relationship	SUP	-
3.4.1 User and System State (context switching, system stack)	SUP	3-24
3.4.1.1 Switching to System State Asynchronously		
(interrupts, interrupt level, \$INTEN)	SUP	3-25
Table 3-2 Values of the Interrupt Level Counter	SUP	3-25
Table 3-3 Job's Stack after \$INTEN	SUP	3-26
Figure 3-17 Interrupts and Execution States	SUP	3-26
3.4.1.2 Switching to System State Synchronously (\$ENSYS)	SUP	-
		-
Table 3-4 Job's Stack after \$ENSYS	SUP	3-27
3.4.1.3 Returning to User State	SUP	
3.6.1.1 Configuration Word (hardware, monitor, clock)	SUP	3-51
Table 3-9 The Configuration Word, Offset 300	SUP	-
3.6.1.4 System Generation Features Word	201	5 51
(I/O timeout, error logging)	SUP	3-54
Table 3-12 System Generation Features Word, Offset 370	SUP	3-54
3.6.3 Queue Element Format Summary	SUP	
3.6.3.1 I/O Queue Element	SUP	
		-
Figure 3-24 I/O Queue Element Format	SUP	
3.6.3.2 Completion Queue Element	SUP	3-60
Figure 3-25 Completion Queue Element Format	SUP	3-60
3.6.3.3 Synch Queue Element	SUP	-
		-
Figure 3-26 Synch Queue Element Format	SUP	
3.6.3.4 Fork Queue Element	SUP	-
Figure 3-27 Fork Queue Element Format	SUP	3-61
3.6.3.5 Timer Queue Element	SUP	
3.6.4 I/O Channel Format (channel status word)	SUP	-
Figure 3-28 Timer Queue Element Format	SUP	3-62
Figure 3-29 I/O Channel Description	SUP	3-62

3.6.5 Device Tables	CIID	2 62
3.6.5.1 \$PNAME Table (permanent names)	SUP SUP	3-62
3.6.5.2 \$STAT Table (device status)	SUP	3-62 3-63
Table 3-16 Channel Status Word (CSW)	SUP	3-63
3.6.5.3 \$DVREC Table (code locations)	SUP	3-03 3-64
3.6.5.4 \$ENTRY Table (entry points)	SUP	3-64
3.6.5.5 \$HSIZE Table (handler size)	SUP	3-64
3.6.5.6 \$DVSIZ Table (device size)	SUP	3-64
J.C.J.C WD.COLL LADIE (GEVICE SIZE)	SUL	5-04
4.2.4.1 Page Address Register (PAR)	SUP	4-13
Figure 4-14 Correspondence Between Pages and Active Page		
Registers	SUP	4-13
Figure 4-15 Page Address Register (PAR)	SUP	4-13
4.2.5 Converting a 16-Bit Address to an 18-Bit Address	SUP	4-14
4.2.7 Kernel and User Processor Modes	SUP	4-16
Figure 4-19 Processor Status Word and Active Page Registers	SUP	4-17
4.6.5 I/O Queue Element (XM)	SUP	4-59
4.8.1 PAR1 Restriction	SUP	4-66
4.8.3 PAR2 Restriction	SUP	4-67
6.2 Teterment Deferre T/O	0.110	6.0
6.2 Interrupt-Driven I/O	SUP	6-2
<ul><li>6.2.1 How an Interrupt Works (interrupt vector, RTI)</li><li>6.2.2 Device and Processor Priorities</li></ul>	SUP	6-3
	SUP SUP	6-3
Figure 6-1 RT-11 Priority Structure 6.2.3 Processor Status (PS) Word	SUP	6-3
6.3 In-Line Interrupt Service Routines Versus Device	SUP	6-4
Handlers	SUP	6-4
Figure 6-2 Processor (PS) Word	SUP	6-5
Figure 6-3 In-Line Interrupt Service Routines and Device	201	
Handlers	SUP	6-7
6.4.1 Get to Know Your Device	SUP	6-8
6.4.2 Study the Structure of an Interrupt Service Routine	SUP	6-10
6.4.3 Study the Skeleton Interrupt Service Routine	SUP	6-10
6.5 Structure of an Interrupt Service Routine	SUP	6-11
6.5.4 Lowering Processor Priority: .INTEN (system state)	SUP	6-13
6.5.5 Issuing Programmed Requests: .SYNCH (user state)	SUP	6-14
6.5.6 Running at Fork Level: .FORK (system state, fork block)		6-15
Table 6-1 Synch Block	SUP	6-15
Table 6-2 Fork Block	SUP	
6.5.7 Summary of .INTEN, .FORK, and .SYNCH Action (registers)	SUP	6-17
Table 6-3         Summary of Interrupt Service Routine Macro	CIID	6 177
Calls	SUP SUP	6-17 6-18
6.5.8 Exiting From Interrupt Service: RTS PC Figure 6-4 Summary of Registers in Interrupt Service	SUL	0-10
Routine Macro Calls	SUP	6-18
6.6 Skeleton Outline of an Interrupt Service Routine	SUP	6-19
Figure 6-5 Skeleton Interrupt Service Routine	SUP	6-19
6.7 Interrupt Service Routines in XM Systems		
(kernel mapping, PAR1, .SYNCH)	SUP	6-20
Figure 6-6 Kernel and Privileged Mapping	SUP	6-21
Figure 6-7 Interrupt Service Routine Mapping Error	SUP	6-22
Figure 6-8 PAR1 Restrictions for Interrupt Service		
Routines	SUP	6-23

7.2Structure of a Device HandlerSUP7-37.2.1Preamble SectionSUP7-37.2.1.1DDDEF Macro (.MCALL, SYSGEN conditionals)SUP7-37.2.1.2Device-Identifier ByteSUP7-6Table 7-1Device-Identifier Byte ValuesSUP7-7Table 7-2Device Status Word (.SPFUN, aborts, internal queuing)SUP7-77.2.1.4Device Size WordSUP7-77.2.1.4Device Size WordSUP7-77.2.1.4Device Size WordSUP7-97.2.2.1Information in Block 0SUP7-97.2.2.2First Five Words of the HandlerSUP7-97.2.2.3JDREG MacroSUP7-97.2.2.4Multi-Vector Handlers: .DRVTB MacroSUP7-10Table 7-4Handler Header WordsSUP7-117.2.2.5S Condition CodesSUP7-117.2.4.1Hotrrupt Service SectionSUP7-147.2.4.1Abort Priority to Device PrioritySUP7-147.2.4.2Lowering the Priority to Device PrioritySUP7-147.2.4.3JDRAST MacroSUP7-167.2.5I/C Completion SectionSUP7-167.2.6.1The JDRED MacroSUP7-187.2.6.2Pseudo-DevicesSUP7-187.2.6.1The JDRED MacroSUP7-167.2.6.2Pseudo-DevicesSUP7-197.3Skeleton Outline of a Device HandlerSUP	<ul> <li>7.1 How to Plan a Device Handler</li> <li>7.1.1 Get to Know Your Device</li> <li>7.1.2 Study the Structure of a Standard Device Handler</li> <li>7.1.3 Study the Skeleton Device Handler</li> <li>7.1.4 Think About Using the Special Features</li> <li>7.1.5 Study the Sample Handlers</li> <li>7.1.6 Prepare a Flowchart of the Device Handler</li> <li>7.1.7 Write the Code (position independent code)</li> <li>7.1.8 Install, Test, and Debug the Handler</li> </ul>	SUP SUP SUP SUP SUP SUP SUP SUP	7-1 7-2 7-2 7-2 7-2 7-2 7-2 7-2
7.2.1.1 .DRDEF Macro (.MCALL, SYSGEN conditionals)SUP7-37.2.1.2 Device-Identifier ByteSUP7-6Table 7-1Device-Identifier Byte ValuesSUP7-77.2.1.3 Device Status Word (.SPFUN, aborts, internal queuing)SUP7-7Table 7-2Device Status WordSUP7-77.2.1.4 Device Size WordSUP7-97.2.2.1Information in Block 0SUP7-97.2.2.1 Information in Block 0SUP7-97.2.2.2 First Five Words of the HandlerSUP7-97.2.2.3 .DRBEG MacroSUP7-10Table 7-4Handler Header WordsSUP7.2.3 .JC Initiation SectionSUP7-107.2.4.1 Abort Entry Point (.FORK)SUP7-117.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP7-167.2.5 I/O Completion SectionSUP7-167.2.6.1 The .DREND MacroSUP7-187.2.6.2 Pseudo-DevicesSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1 Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7.4Handlers that Queue InternallySUP7-227.4.3 Abort Procedures for Handlers that Queue InternallySUP7-227.5.1 How the SET Co	7.2 Structure of a Device Handler	SUP	7-3
7.2.1.2 Device-Identifier ByteSUP7-6Table 7-1Device-Identifier Byte ValuesSUP7-77.2.1.3 Device Status Word (.SFFUN, aborts, internal queuing)SUP7-7Table 7-2Device Status WordSUP7-77.2.1.4 Device Size WordSUP7-77.2.2.4 Header SectionSUP7-9Table 7-3Information in Block 0SUP7-97.2.2.4 Header SectionSUP7-97.2.2.5 First Five Words of the HandlerSUP7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP7-107.2.2.5 PS Condition CodesSUP7-107.2.4.1 Abort Entry Point (.FORK)SUP7-117.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP7-167.2.6.1 The .DREND MacroSUP7-167.2.6.2 Fseudo-DevicesSUP7-187.2.6.3 It andler Termination SectionSUP7-187.2.6.4 Handler Termination SectionSUP7-187.2.6.5 Pseudo-DevicesSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-207.4.4 Handlers that Queue InternallySUP7-207.4.5 Abort Frocedures for Handlers that Queue InternallySUP7-227.4.6.2 Fseudo-Device for Handlers that Queue InternallySUP7-227.4.4 Boot Frocedures for Handlers that Queue InternallySUP7-227.4.5 Abort Frocedures for Ha	7.2.1 Preamble Section	SUP	7-3
Table 7-1Device-Identifier Byte ValuesSUP7-67.2.1.3 Device Status Word (.SFFUN, aborts, internal queuing)SUP7-7Table 7-2Device Status WordSUP7-77.2.1.4 Device Size WordSUP7-77.2.2.4 Header SectionSUP7-97.2.2.4 Header SectionSUP7-97.2.2.5 Header SectionSUP7-97.2.2.4 Header SectionSUP7-97.2.2.5 A MBEG MacroSUP7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP7-107.2.2.5 PS Condition CodesSUP7-117.2.3 I/O Initiation SectionSUP7-117.2.4 Interrupt Service SectionSUP7-147.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP7-157.2.6.4 Handler Termination SectionSUP7-167.2.6.4 Handler Termination SectionSUP7-187.2.6.4 Handler Termination SectionSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-197.4 Handlers that Queue InternallySUP7-207.4.1 Implementing Internal QueuingSUP7-217.4.2Interrupt Service for Handlers that Queue InternallySUP7.4.3 Abort Procedures for Handlers that Queue InternallySUP7.5SET OptionsSUP7-227.5.1 How the SET Command ExecutesSUP7-237.5.3.DRSET MacroSUP7-23	7.2.1.1 .DRDEF Macro (.MCALL, SYSGEN conditionals)	SUP	7-3
7.2.1.3 Device Status Word (.SPFUN, aborts, internal queuing)SUP7-7Table 7-2Device Status WordSUP7-77.2.1.4 Device Size WordSUP7-87.2.2 Header SectionSUP7-97.2.2.1 Information in Block 0SUP7-97.2.2.2 First Five Words of the HandlerSUP7-97.2.2.3 .DRBEG MacroSUP7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP7-107.2.2.5 FS Condition CodesSUP7-117.2.4.1 Abort Entry Point (.FORK)SUP7-137.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP7-167.2.5 I/O Completion SectionSUP7-167.2.6.1 The .DREND MacroSUP7-187.2.6.2 Pseudo-DevicesSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1 Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7.4Handlers that Queue InternallySUP7.5SET OptionsSUP7-227.5.1 How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-237.5.4SET Table FormatSUP7-24	7.2.1.2 Device-Identifier Byte		7-6
Table 7-2Device Status WordSUP7-77.2.1.4 Device Size WordSUP7-87.2.2. Header SectionSUP7-97.2.2.1 Information in Block 0SUP7-9Table 7-3Information in Block 0SUP7.2.2.2 First Five Words of the HandlerSUP7-97.2.2.3 DREEG MacroSUP7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP7-10Table 7-4Handler Header WordsSUP7.2.2.5 FS Condition CodesSUP7-117.2.4 Interrupt Service SectionSUP7-147.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.2 Lowering the Priority to Device PrioritySUP7-157.2.5 I/O Completion SectionSUP7-167.2.6.4 Handler Termination SectionSUP7-187.2.6.5 I/O Completion SectionSUP7-187.2.6.6 Handler Termination SectionSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-197.4 Handlers that Queue InternallySUP7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP7-227.4.3 Abort Procedures for Handlers that Queue InternallySUP7-227.5 SET OptionsSUP7-227-227.5.1 How the SET Command ExecutesSUP7-237.5.2 SET Table FormatSUP7-237.5.3 DRSET MacroSUP7-237.5.4 SET Table FormatSUP7-24	Table 7-1 Device-Identifier Byte Values	SUP	7-6
7.2.1.4 Device Size WordSUP7-87.2.2 Header SectionSUP7-97.2.2.1 Information in Block 0SUP7-97.2.2.2 First Five Words of the HandlerSUP7-97.2.2.3 .DRBEG MacroSUP7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP7-97.2.2.5 PS Condition CodesSUP7-107.2.4.1 Interrupt Service SectionSUP7-117.2.3 .JORBEG MacroSUP7-117.2.4.4 Interrupt Service SectionSUP7-117.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP7-157.2.5 I/O Completion SectionSUP7-167.2.6.1 The .DREND MacroSUP7-187.2.6.2 Pseudo-DevicesSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-197.4 Handlers that Queue InternallySUP7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP7-217.4.3 Abort Procedures for Handlers that Queue InternallySUP7-227.5.1 How the SET Command ExecutesSUP7-237.5.2 SET OptionsSUP7-237.5.3 .DRST MacroSUP7-237.5.4 Kow the SET Command ExecutesSUP7-237.5.5 SET OptionsSUP7-237.5.6.2 Net OptionsSUP7-237.5.7 Kow the SET Command ExecutesSUP7-237.5.8 SU Opti			
7.2.2Header SectionSUP7-97.2.2.1Information in Block 0SUP7-9Table 7-3Information in Block 0SUP7-97.2.2.2First Five Words of the HandlerSUP7-97.2.2.3JRBEG MacroSUP7-97.2.2.4Multi-Vector Handlers: .DRVTB MacroSUP7-10Table 7-4Handler Header WordsSUP7-117.2.2.5PS Condition CodesSUP7-117.2.3I/O Initiation Section (system state)SUP7-117.2.4Interrupt Service SectionSUP7-147.2.4.1Abort Entry Point (.FORK)SUP7-147.2.4.2Lowering the Priority to Device PrioritySUP7-147.2.4.4Guidelines for Coding the Interrupt Service SectionSUP7-157.2.5I/O Completion SectionSUP7-167.2.6.1The .DREND MacroSUP7-187.2.6.2Pseudo-DevicesSUP7-197.3Skeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-227.4.2Interrupt Service for Handlers that Queue InternallySUP7.4.3Abort Procedures for Handlers that Queue InternallySUP7.4.4Handlers thermal ExecutesSUP7-227.5.1How the SET Command ExecutesSUP7-227.5.1How the SET Command ExecutesSUP7-23			
7.2.2.1 Information in Block 0SUP7-9Table 7-3Information in Block 0SUP7-97.2.2.2 First Five Words of the HandlerSUP7-97.2.2.3 DRBEG MacroSUP7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP7-10Table 7-4Handler Header WordsSUP7-107.2.2.5 PS Condition CodesSUP7-117.2.4.1 Abort Entry Point (.FORK)SUP7-117.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP7-157.2.5 I/O Completion SectionSUP7-167.2.6.1 The .DREND MacroSUP7-167.2.6.2 Pseudo-DevicesSUP7-187.2.6.3 Exeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-197.4.1 Implementing Internal QueuingSUP7-207.4.2Abort Procedures for Handlers that Queue InternallySUP7.5SET OptionsSUP7-227.5.1 How the SET Command ExecutesSUP7-227.5.1 How the SET Command ExecutesSUP7-237.5.3 .DRSET MacroSUP7-23			
Table 7-3Information in Block 0SUP7-97.2.2.2 First Five Words of the HandlerSUP7-97.2.2.3 .DRBEG MacroSUP7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP7-10Table 7-4Handler Header WordsSUP7-117.2.2.5 PS Condition CodesSUP7-117.2.4.1 Mather Entry Point (.FORK)SUP7-147.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP(.FORK, retries)SUP7-167.2.5I/O Completion SectionSUP7.2.6.1 The .DREND MacroSUP7-187.2.6.2 Pseudo-DevicesSUP7-197.3Skeleton Outline of a Device HandlerSUP7.4Handlers that Queue InternallySUP7.4.1Implementing Internal QueuingSUP7.4.2Interrupt Service for Handlers that Queue InternallySUP7.4.3Abort Procedures for Handlers that Queue InternallySUP7.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.3.DRSET MacroSUP7-237.5.3.DRSET MacroSUP7-24			
7.2.2.2 First Five Words of the HandlerSUP 7-97.2.2.3 .DRBEG MacroSUP 7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP 7-10Table 7-4Handler Header WordsSUP 7-107.2.2.5 PS Condition CodesSUP 7-117.2.3 I/O Initiation Section (system state)SUP 7-117.2.4 Interrupt Service SectionSUP 7-137.2.4.1 Abort Entry Point (.FORK)SUP 7-147.2.4.2 Lowering the Priority to Device PrioritySUP 7-147.2.4.3 .DRAST MacroSUP 7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP 7-157.2.5 I/O Completion SectionSUP 7-167.2.6.1 The .DREND MacroSUP 7-187.2.6.2 Pseudo-DevicesSUP 7-197.3 Skeleton Outline of a Device HandlerSUP 7-197.4 Handlers that Queue InternallySUP 7-207.4.1 Implementing Internal QueuingSUP 7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP 7-227.5.1 How the SET Command ExecutesSUP 7-237.5.3 .DRSET MacroSUP 7-237.5.3 .DRSET MacroSUP 7-23			
7.2.2.3 .DRBEG MacroSUP 7-97.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP 7-10Table 7-4Handler Header WordsSUP 7-107.2.2.5 PS Condition CodesSUP 7-117.2.3 I/O Initiation Section (system state)SUP 7-117.2.4 Interrupt Service SectionSUP 7-147.2.4.1 Abort Entry Point (.FORK)SUP 7-147.2.4.2 Lowering the Priority to Device PrioritySUP 7-147.2.4.3 .DRAST MacroSUP 7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP 7-157.2.5 I/O Completion SectionSUP 7-167.2.6.1 The .DREND MacroSUP 7-187.2.6.2 Pseudo-DevicesSUP 7-197.3 Skeleton Outline of a Device HandlerSUP 7-197.4.1 Implementing InternallySUP 7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP 7-207.4.3 Abort Procedures for Handlers that Queue InternallySUP 7-227.5 SET OptionsSUP 7-237.5.3 .DRSET MacroSUP 7-237.5.3 .DRSET MacroSUP 7-24			
7.2.2.4 Multi-Vector Handlers: .DRVTB MacroSUP 7-10Table 7-4Handler Header WordsSUP 7-107.2.2.5 PS Condition CodesSUP 7-117.2.3 I/O Initiation Section (system state)SUP 7-117.2.4 Interrupt Service SectionSUP 7-147.2.4.1 Abort Entry Point (.FORK)SUP 7-147.2.4.2 Lowering the Priority to Device PrioritySUP 7-147.2.4.3 .DRAST MacroSUP 7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP 7-157.2.5 I/O Completion SectionSUP 7-167.2.6.1 The .DREND MacroSUP 7-187.2.6.2 Pseudo-DevicesSUP 7-197.3 Skeleton Outline of a Device HandlerSUP 7-197.4 Handlers that Queue InternallySUP 7-207.4.1 Implementing Internal QueuingSUP 7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP 7-227.5 SET OptionsSUP 7-227.5.1 How the SET Command ExecutesSUP 7-237.5.3 .DRSET MacroSUP 7-237.5.3 .DRSET MacroSUP 7-24			
Table 7-4Handler Header WordsSUP7-107.2.2.5 PS Condition CodesSUP7-117.2.3 I/O Initiation Section (system state)SUP7-117.2.4Interrupt Service SectionSUP7-117.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP(.FORK, retries)SUP7-167.2.6Handler Termination SectionSUP7.2.6.1 The .DREND MacroSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-19Figure 7-1Skeleton Device HandlerSUP7.4Handlers that Queue InternallySUP7.4.1 Implementing Internal QueuingSUP7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP7.4.3 Abort Procedures for Handlers that Queue InternallySUP7.5.4SET OptionsSUP7.5.2 SET Table FormatSUP7.5.3.DRSET MacroSUP7.5.3.DRSET MacroSUP			
7.2.2.5 PS Condition CodesSUP7-117.2.3 I/O Initiation Section (system state)SUP7-117.2.4 Interrupt Service SectionSUP7-137.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.3 JRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service SectionSUP(.FORK, retries)SUP7-157.2.5 I/O Completion SectionSUP(.channel status word, end-of-file)SUP7.2.6.1 The .DREND MacroSUP7.2.6.2 Pseudo-DevicesSUP7.3 Skeleton Outline of a Device HandlerSUP7.4 Handlers that Queue InternallySUP7.4.1 Implementing Internal QueuingSUP7.4.2 Interrupt Service for Handlers that Queue InternallySUP7.4.3 Abort Procedures for Handlers that Queue InternallySUP7.5 SET OptionsSUP7.5.1 How the SET Command ExecutesSUP7.5.3 .DRSET MacroSUP7.5.3 .DRSET MacroSUP			
7.2.3I/O Initiation Section (system state)SUP7-117.2.4Interrupt Service SectionSUP7-137.2.4.1Abort Entry Point (.FORK)SUP7-147.2.4.2Lowering the Priority to Device PrioritySUP7-147.2.4.3JRAST MacroSUP7-147.2.4.4Guidelines for Coding the Interrupt Service SectionSUP7-157.2.5I/O Completion SectionSUP7-167.2.6Handler Termination SectionSUP7-187.2.6.1The .DREND MacroSUP7-197.3Skeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-217.4.2Interrupt Service for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24			
7.2.4Interrupt Service SectionSUP7-137.2.4.1Abort Entry Point (.FORK)SUP7-147.2.4.2Lowering the Priority to Device PrioritySUP7-147.2.4.3.DRAST MacroSUP7-147.2.4.4Guidelines for Coding the Interrupt Service SectionSUP7-147.2.5I/O Completion SectionSUP7-167.2.6Handler Termination SectionSUP7-187.2.6.1The .DREND MacroSUP7-197.3Skeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24			
7.2.4.1 Abort Entry Point (.FORK)SUP7-147.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service Section (.FORK, retries)SUP7-157.2.5 I/O Completion Section (channel status word, end-of-file)SUP7-167.2.6 Handler Termination Section (channel status word, end-of-file)SUP7-187.2.6.1 The .DREND MacroSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-197.4 Handlers that Queue InternallySUP7-207.4.1 Implementing Internal QueuingSUP7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP7-217.4.3 Abort Procedures for Handlers that Queue InternallySUP7-227.5 SET OptionsSUP7-237.5.2 SET Table FormatSUP7-237.5.3 .DRSET MacroSUP7-23			
7.2.4.2 Lowering the Priority to Device PrioritySUP7-147.2.4.3 .DRAST MacroSUP7-147.2.4.4 Guidelines for Coding the Interrupt Service Section (.FORK, retries)SUP7-157.2.5 I/O Completion Section (channel status word, end-of-file)SUP7-167.2.6 Handler Termination SectionSUP7-187.2.6.1 The .DREND MacroSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-197.4 Handlers that Queue InternallySUP7-207.4.1 Implementing Internal QueuingSUP7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP7-217.4.3 Abort Procedures for Handlers that Queue InternallySUP7-227.5 SET OptionsSUP7-237.5.2 SET Table FormatSUP7-237.5.3 .DRSET MacroSUP7-23			
7.2.4.4 Guidelines for Coding the Interrupt Service Section (.FORK, retries)SUP 7-157.2.5 I/O Completion Section (channel status word, end-of-file)SUP 7-167.2.6 Handler Termination Section 7.2.6.1 The .DREND Macro 7.2.6.2 Pseudo-DevicesSUP 7-187.3 Skeleton Outline of a Device Handler Figure 7-1SUP 7-197.4 Handlers that Queue Internally 7.4.1 Implementing Internal Queuing 7.4.2 Interrupt Service for Handlers that Queue Internally 7.4.3 Abort Procedures for Handlers that Queue Internally SUP 7-22SUP 7-227.5 SET Options 7.5.1 How the SET Command Executes 7.5.2 SET Table Format 7.5.3 .DRSET MacroSUP 7-23 SUP 7-24		SUP	7-14
(.FORK, retries)SUP7-157.2.5I/O Completion Section (channel status word, end-of-file)SUP7-167.2.6Handler Termination SectionSUP7-187.2.6.1The .DREND MacroSUP7-187.2.6.2Pseudo-DevicesSUP7-197.3Skeleton Outline of a Device HandlerSUP7-19Figure 7-1Skeleton Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-217.4.2Interrupt Service for Handlers that Queue InternallySUP7-227.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-237.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24	7.2.4.3 .DRAST Macro	SUP	7-14
7.2.5I/O Completion Section (channel status word, end-of-file)SUP7.2.6Handler Termination SectionSUP7.2.6.1The .DREND MacroSUP7.2.6.2Pseudo-DevicesSUP7.3Skeleton Outline of a Device HandlerSUP7.4Handlers that Queue InternallySUP7.4.1Implementing Internal QueuingSUP7.4.2Interrupt Service for Handlers that Queue InternallySUP7.4.3Abort Procedures for Handlers that Queue InternallySUP7.5SET OptionsSUP7.5.1How the SET Command ExecutesSUP7.5.2SET Table FormatSUP7.5.3.DRSET MacroSUP		SUP	7-15
(channel status word, end-of-file)SUP7-167.2.6Handler Termination SectionSUP7-187.2.6.1The .DREND MacroSUP7-187.2.6.2Pseudo-DevicesSUP7-197.3Skeleton Outline of a Device HandlerSUP7-197.3Skeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-237.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24	7.2.5 I/O Completion Section	001	1 15
7.2.6Handler Termination SectionSUP7-187.2.6.1The .DREND MacroSUP7-187.2.6.2Pseudo-DevicesSUP7-197.3Skeleton Outline of a Device HandlerSUP7-197.3Skeleton Outline of a Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-237.5.1How the SET Command ExecutesSUP7-237.5.3.DRSET MacroSUP7-23		SUP	7-16
7.2.6.2 Pseudo-DevicesSUP7-197.3 Skeleton Outline of a Device HandlerSUP7-19Figure 7-1Skeleton Device HandlerSUP7-197.4 Handlers that Queue InternallySUP7-207.4.1 Implementing Internal QueuingSUP7-207.4.2 Interrupt Service for Handlers that Queue InternallySUP7-217.4.3 Abort Procedures for Handlers that Queue InternallySUP7-227.5 SET OptionsSUP7-227.5.1 How the SET Command ExecutesSUP7-237.5.2 SET Table FormatSUP7-237.5.3 .DRSET MacroSUP7-24		SUP	
7.3Skeleton Outline of a Device HandlerSUP SUP 7-19Figure 7-1Skeleton Device HandlerSUP SUP 7-197.4Handlers that Queue Internally Implementing Internal QueuingSUP SUP 7-207.4.1Implementing Internal Queuing 7.4.2SUP Interrupt Service for Handlers that Queue Internally SUP 7.4.3SUP Abort Procedures for Handlers that Queue Internally SUP 7-227.5SET Options 7.5.1SUP How the SET Command Executes SUP 7.5.3SUP .7-23 SUP SUP 7-23 SUP SUP 7-24	7.2.6.1 The .DREND Macro	SUP	7-18
Figure 7-1Skeleton Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24	7.2.6.2 Pseudo-Devices	SUP	7-19
Figure 7-1Skeleton Device HandlerSUP7-197.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24	7.3 Skeleton Outline of a Device Handler	SUP	7-19
7.4Handlers that Queue InternallySUP7-207.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24			
7.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24			
7.4.1Implementing Internal QueuingSUP7-207.4.2Interrupt Service for Handlers that Queue InternallySUP7-217.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24	7.4 Handlers that Queue Internally	SUP	7-20
7.4.3Abort Procedures for Handlers that Queue InternallySUP7-227.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24		SUP	7-20
7.5SET OptionsSUP7-227.5.1How the SET Command ExecutesSUP7-237.5.2SET Table FormatSUP7-237.5.3.DRSET MacroSUP7-24	7.4.2 Interrupt Service for Handlers that Queue Internally	SUP	7-21
7.5.1How the SET Command ExecutesSUP 7-237.5.2SET Table FormatSUP 7-237.5.3.DRSET MacroSUP 7-24	7.4.3 Abort Procedures for Handlers that Queue Internally	SUP	7-22
7.5.1How the SET Command ExecutesSUP 7-237.5.2SET Table FormatSUP 7-237.5.3.DRSET MacroSUP 7-24	7.5 SET Options	SIIP	7-22
7.5.2         SET Table Format         SUP 7-23           7.5.3         .DRSET Macro         SUP 7-24	-		
7.5.3 .DRSET Macro SUP 7-24			
Figure 7-2 SET Option Table SUP 7-24			
7.5.4 Routines to Modify the Handler SUP 7-25			
7.5.5 Examples of SET Options SUP 7-25	7.5.5 Examples of SET Options	SUP	7-25

7.6 Device I/O Timeout	SUP	7-28
7.6.1 .TIMIO Macro (.FORK, kernel mapping)	SUP	7-28
Table 7-5 Timer Block Format	SUP	
7.6.2 .CTIMIO Macro (.FORK, abort)		7-30
7.6.3 Device Time-out Applications		7-31
7.6.3.1 Multi-terminal Services	SUP	7-31
7.6.3.2 Typical Timer Procedure for a Disk Handler		
(system stack)	SUP	7-31
7.6.3.3 Line Printer Handler Example		
		7-32
Figure 7-3 Line Printer Handler Example	SUP	7-33
7.7 Error Logging	SUP	7-34
7.7.1 When and How to Call the Error Logger (.FORK)		7-35
7.7.1.1 To Log a Successful Transfer	SUP	
		7-35
7.7.1.2 To Log a Hard Error	SUP	· +
7.7.1.3 To Log a Soft Error	SUP	7-35
7.7.1.4 Differences Between Hard and Soft Errors	SUP	7-36
7.7.1.5 To Call the Error Logger		7-36
7.7.2 Error Logging Examples		7-37
7.7.3 How to Add a Device to the Reporting Program	SUP	7-37
7.8 Special Functions	SUP	7-38
7.8.1 .SPFUN Programmed Request	SUP	
		. +
7.8.2 How to Support Special Functions in a Device Hand		
7.8.3 Variable Size Volumes	SUP	7-40
7.8.4 Bad Block Replacement	SUP	7-40
7.8.5 Devices with Special Directories	SUP	7-40
7.9 Device Handlers in XM Systems	SUP	7-41
7.9.1 Naming Conventions and the System Conditional	SUP	
7.9.2 XM Environment (PAR1, PAR2, kernel mapping)	SUP	7-41
7.9.3 The Queue Element in XM	SUP	7-42
Figure 7-4 Device Handler in XM	SUP	7-43
7.9.4 DMA Devices: \$MPPHY Routine	SUP	7-44
7.9.5 Character Devices: \$GETBYT and \$PUTBYT Routines	SUP	
7.9.5.1 \$GETBYT Routine	SUP	7-45
7.9.5.2 \$PUTBYT Routine	SUP	7-45
7.9.6 Any Device: \$PUTWRD Routine	SUP	
7.9.7 Handlers That Access the User Buffer Directly (PA		
Figure 7-5 Device Handler Mapping to User Buffer Are		
Figure 7-6 PAR1 Mapping	SUP	7-49
		-
7.10 Tystem Device Handlers and Bootstraps	SUP	7-50
7.10.1 Monitor Files	SUP	7-50
7.10.2 Creating a System Device Handler	SUP	7-51
	SUP	7-51
7.10.2.1 Primary Driver	SUP	
7.10.2.2 Entry Routine		
7.10.2.3 Software Bootstrap	SUP	7-52
7.10.2.4 Bootstrap Read Routine	SUP	7-52
7.10.2.5 Bootstrap Error Routine	SUP	7-52
	SUP	
7.10.2.6 .DRBOT Macro		
7.10.3 DUP and the Bootstrap Process	SUP	7-53
7.10.3.1 BOOT ddn:filnam	SUP	
7.10.3.2 COPY/BOOT xxn:filnam ddm:	SUP	7-54
Table 7-6 DUP Information	SUP	7-54
	SUP	
	SUP	
Figure 7-7 BOOT ddn:filnam Procedure		
Figure 7-8 COPY/BOOT xxn:filnam ddm: Procedure	SUP	
Table 7-7 DUP Information	SUP	•
Figure 7-9 BOOT ddn: Procedure	SUP	7-57

7.11.1 Assembl 7.11.2 Linking 7.11.3 Install	Assemble, Link, and Install a Device Handler ing a Device Handler a Device Handler ing a Device Handler	SUP SUP SUP SUP	7-57 7-57 7-58 7-58
7.11.3.1	Using the Bootstrap to Install Handlers Automatically	SUP	7-58
7.11.3.2	Using the INSTALL Command to Install Handlers	001	1-50
	Manually	SUP	7-59
7.11.3.3	Using the DEV Macro to Aid Automatic Installation	SUP	7-60
Figure 7-10	Bootstrap Algorithm for Installing Device	CUID	7 60
	Handlers	SUP	7-60
Figure 7-11	Installing a New Device Handler	SUP	7-61
7.11.3.4	Installing Devices Whose Hardware Is Present	SUP	7-62
7.11.3.5	Writing an Installation Verification Routine	SUP	7-62
7.11.3.6	Overriding the Hardware Restriction	SUP	7-65
7.12.1 Using C	Test and Debug a Device Handler DT to Test a Handler ODT and a Device Handler in Memory DT in XM	SUP SUP SUP SUP	7-65 7-66 7-67 7-68

Russell L. Morrison II Plessey Peripheral Systems P.O. Box 19616 Irvine, CA 92714

#### (714)540-9945

I have discovered a small problem with RT-11 V5 and RX03 dual sided, dual density floppies: the new version has all the code that used to support RX03's deleted. What used to be a relatively simple patch to enable RX03 support is now a rather involved patch which re-enters all the old RX03 support code.

If you would like a copy of this patch please request in writing to the editor of the "Mini-Tasker".

Sincerely,

th

Russell L. Morrison II Systems Analyst, Software Support

\*\*\*\*\*\*\*

HIGH LEVEL MULTILANGUAGE MACHINE-INDEPENDENT PROGRAMMATION (16, 32, 36, ... BITS) : A SUBROUTINE FOR BIT MANIPULATIONS IN BASIC AND FORTRAN IV.

\*\*\*\*\*\*\*\*\*\*\*

BY DANIEL GUINIER

LABORATOIRE DE PHYSIOLOGIE COMPAREE DES REGULATIONS GROUPE DE LABORATOIRES DU CNRS DE STRASBOURG-CRONENBOURG 23 RUE DU LOESS B. P. 20 CR 67037 STRASBOURG CEDEX, FRANCE

INTRODUCTION : \*\*\*\*\*\*\*

MANIPULATION OR EXAMINATION OF BITS OF A MEMORY WORD PERMITS COMPRESSION OF BINARY DATA THAT CAN REACH A VERY INTERESTING RATE FOR STORAGE, CODING OR DATA ACQUISITION. THIS ALSO ALLOWS LOGICAL OPERATIONS APART FROM USING MACHINE CODE OR ASSEMBLER LANGUAGE WHICH ARE PARTICULAR TO A GIVEN COMPUTER.

WE HAVE REALIZED A SUBROUTINE AND ITS CALLING PROGRAM WRITTEN AS IN FORTRAN IV AND ALSO IN BASIC (TESTED WITH ZX81 SINCLAIR WHICH IS THE LEAST EXPENSIVE MODEL IN THE MARKET OF MICRO-COMPUTERS). OUR PURPOSE IS TO USE THIS METHOD ON ALL TYPES AND ORGANIZATIONS OF COMPUTERS (16, 32, 36, ... BITS) AS WELL AS TO COMPARE THESE TWO LANGUAGES.

> METHODS : \*\*\*\*\*\*\*\*\* REPRESENTATION OF INTEGERS : -----

AN INTEGER I IS STORED IN A MEMORY WORD OF 16, 32, 36, ... BITS WHOSE THE HIGHEST WEIGHTED BIT IS THE BIT OF SIGN; IF THIS BIT IS RESET TO ZERO, THE NUMBER IS POSITIVE, OTHERWISE, IT IS POSITIVE.

EXAMPLE :

- LET BIT(J)=0 OR 1. THE ACTUAL VALUE OF THE J TH. BIT IN THE WORD. IF NBRBIT=16 -(2\*\*15-1) (= I (= 2\*\*15-1 THAT IS -32767 (= I (= 32767 IF NBRBIT=32 -(2\*\*31-1) <= I <= 2\*\*31-1 IF NBRBIT=36 -(2\*\*35-1) (= I (= 2\*\*35-1 BINARY - DECIMAL CONVERSION : THE VALUE OF A MACHINE WORD WHICH IS THE IMAGE OF A FIELD INTEGER I CAN BE EXPRESSED AS : I= BIT(1) + BIT(2)\*2 + BIT(3)\*2\*\*2 + BIT(4)\*2\*\*3 + BIT(NBRBIT-1)\*2\*\*(NBRBIT-2) THE NERBIT-TH BIT GIVES THE SIGN OF I EXAMPLE : 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 ------I I OI OI OI OI OI OI OI OI 11 OI OI OI 11 11 IF WE APPLY WHAT WAS DESCRIBED ABOVE : I = 1 + 1\*2 + 0\*2\*\*2 + ... + 1\*2\*\*6 + ... = 1 + 2 + 2\*\*6 = 1 + 2 + 64 = 67I = + 67 BECAUSE BIT(NBRBIT) = BIT(16) = 0 THIS IS A BINARY - DECIMAL CONVERSION. DECIMAL - BINARY CONVERSION : FOR A POSITIVE INTEGER I, THAT IS AN I WITHOUT ITS SIGN BIT, CAN BE CONVERTED INTO BINARY REPRESENTATION CONTAINED IN THE ELEMENTS BIT(J) OF AN ARRAY BIT( ) THAT ARE THE RESIDUALS OF SUCCESSIVE DIVISIONS PER TWO. EXAMPLE : - TAKE THE ABSOLUE VALUE OF I = 67 67 / 2 = 33 ; RESIDUAL : BIT(1) = 1 33 / 2 = 16 ; RESIDUAL : BIT(2) = 1 16 / 2 = 8 ; RESIDUAL : BIT(3) = 0 8 / 2 = 4 ; RESIDUAL : BIT(4) = 0 4 / 2 = 2; RESIDUAL : BIT(5) = 0 2 / 2 = 1 ; RESIDUAL : BIT(6) = 0 1 / 2 = 0; RESIDUAL : BIT(7) = 1 ALL OTHER BITS FROM BIT(8) TO BIT(NBRBIT-1), THAT IS BIT(15) ARE RESET TO ZERO AND BIT(NBRBIT) = 0 IF I IS POSITIVE. THIS IS A DECIMAL - BINARY CONVERSION.

- LET J, THE INDEX FOR THE POSITION OF THE BITS IN THE WORD (J=1 TO NBRBIT).

- LET NBRBIT=16, 32 OR 36, THE NUMBER OF BITS PER WORD.

FOR NEGATIVE NUMBERS :

IN THE PRECEDING CONVERSIONS, WE WORKED ON POSITIVE INTEGERS TO AVOID TWO'S COMPLEMENTATION, FORM IN WHICH NEGATIVE INTEGERS ARE USUALY STORED, THIS OFERATION WAS AUTOMATICALLY DONE BY A SINGLE INSTRUCTION OF SIGN CHANGE AND COMPLEMENTATION BY I = I (+/-) 2\*\*(NBRBIT-1)WHEN BIT(NBRBIT) = 1.

EXAMPLE :

				1	1.6	1	1.5	5	1.	4	1	3	1	2		1. 1		1.	Ø		9			8		7		6		5			4			ŝ	2		1	
T	101	÷	67	 I	6	1		31		31		0	I 	0	I	•	† I		0	I	0	I 	-	01	[	1	I 	0	I	0	I	-	0	I	6	) I	 1	I	 11	- -
I	H		67	 I		I	1	LI	-	1 I	-	1.	I	1	I		. I	-	1	I	1.	I	-	11	 [ 	0	I	1	I	1	I	-	1	I		. I	 8	- I -	 11	-

```
NUMERICAL EXAMPLES :
******
1ST. EXAMPLE : *****************
*******
INPUT :
I = 67
I1= 0
OUTPUT :
67
*****
INPUT :
I =-67
11= 0
OUTFUT :
-67
1111111110111101
3TH. EXAMPLE : *******************
******
INPUT :
I = 67
11 = 1
BIT 1 = 1
            BIT 9 = 1
BIT 2 = 0
             BIT10 = 1
BIT 3 = 1
             BIT11 = 1
BIT 4 = 1
             BIT12 = 1
BIT 5 = 1
             BIT13 = 1
BIT 6 = 1
             BIT14 = 1
BIT 7 = 0
             BIT15 = 1
BIT 8 = 1
            BIT16 = 1
OUTPUT :
-67
1111111110111101
```

4TH. EXAMPLE : \* \*\*\*\*\*\* INPUT :  $I_{.} = 67$ I1 = 1BIT 1 = 2BIT 9 = 1BIT 2 = 2BIT10 = 1BIT 3 = 2BIT11 = 1BIT 4 = 2BIT12 = 1BIT 5 = 2BIT13 = 1BIT 6 = 2BIT14 = 1BIT 7 = 2BIT15 = 1BIT 8 = 2BIT16 = 1OUTPUT : -1891111111101000011

\*\*\*\*\*\*\*

LISTINGS : \*\*\*\*\*\*

SUBROUTINE BITØ1 INCLUDES THREE PRINCIPAL PHASES : SEARCH OF THE BITS' LEVEL FOR THE FIELD INTEGER I. POSSIBLE CHANGES OF THIS LEVEL (Ø OR 1). RELEASING OF A NEW FIELD INTEGER I AFTER A CHANGE OF LEVEL OF THE BITS.

THESE PHASES ARE IMPLICIT FOR THE TWO VERSIONS (BASIC AND FORTRAN). THE LISTINGS BELOW INCLUDE THE TWO VERSIONS WITH MAINS AND SUBROUTINES AND ALSO FOUR NUMERICAL EXAMPLES.

INPUT AND OUTPUT ARGUMENTS :

FORTRAN IV	BASIC	TYPE	FUNCTION
I	I	IN/OUT	INTEGER FOR THE LEVEL OF THE BITS OF THE ARRAY BIT( ), UNCHANGED FOR OUTPUT IF IND OR I1=0
IND	I 1.	IN	IF IND OR I1=0 : EXAMINATION OF THE BITS' LEVEL, OTHERWISE POSSIBLE CHANGE OF THESE LEVELS.
BIT( )	A( )	OUT	INTEGER ARRAY [0,1], LOADED WITH THE BITS Level of the integer field I.
BITSET( )	B( )	IN	INTEGER ARRAY, GIVING THE BITS' LEVEL TO CHANG FOR ALL BITSET( ) INCLUDED IN [0,1]
NBRBIT	NØ	IN	NUMBER OF BITS IN A MACHINE WORD (NBRBIT=16, 32, 36, BITS).

\* \* LISTING OF THE FORTRAN IV VERSION \* \*\*\*\*\*\* \*\*\*\*\*\* \* MAIN PROGRAM FORTRAN IV \* \*\*\*\*\*\* INTEGER BIT(36), BITSET(36) DATA NBRBIT/16/LEC, IMP/5, 7/ DO 1 J=1, NBRBIT BITSET(J)=-1 1 WRITE(IMP, 2) FORMAT('\$I= ') 2 READ(LEC, 3)I 7 FORMAT(15) WRITE(IMP, 4) FORMAT('\$IND= ') 4 READ(LEC, 3)IND IF(IND. EQ. 0)60 TO 7 DO 6 J=1, NBRBIT WRITE(IMP, 5)J 5 FORMAT('\$BIT', I2,' = ') READ(LEC, 3)K IF(K. GT. 1. OR. K. LT. 0) GO TO 6 BITSET(J)=K CONTINUE 6 C CALL SUBROUTINE BIT01 7 CALL BITØ1(I, IND, BIT, BITSET, NBRBIT) C OUTPUT RESULTS. WRITE(IMP, 8)I, (BIT(K), K=NBRBIT, 1, -1) 8 FORMAT(/18//,2%,1611//) STOP END \*\*\*\* \* SUBROUTINE FORTRAN IV \* \* C C SUBROUTINE BIT01(I, IND, BIT, BITSET, NBRBIT) C C DANIEL GUINIER (1983) C.N.R.S. STRASBOURG C C SUBROUTINE FORTRAN IV FOR EXAMINATION AND/OR CHANGE OF THE VALUE C OF ONE OR SEVERAL BITS IN A MACHINE WORD FOR ANY TYPE OF COMPUTER C ARCHITECTURE (16, 32, 36, ... BITS).

```
INTEGER BIT(36), BITSET(36)
       NBR=NBRBIT-1
       IP=I
       BIT(NBRBIT)=0
       IF(IP GE. 0)GO TO 1
       BIT(NBRBIT)=1
       IP=IP+2. **(NBRBIT-1)
1
       DO 2 J=1, NBR
       BIT(J)=MOD(IP,2)
2
       IP=IP/2
       IF(IND. EQ. 0)RETURN
       DO 4 J=1, NBRBIT
       IF(BITSET(J), NE. 0)GO TO 3
       BIT (J) = 0
       BITSET(J) = -1
       GO TO 4
       IF(BITSET(J), NE. 1)60 TO 4
3
       BIT (J)= 1
       BITSET(J)=-1
4
       CONTINUE
       I=BIT(NBR)
       N=NBR-1
       DO 5 J=N, 1, -1
5
       I = I * 2 + BIT(J)
       IF(BIT(NBRBIT).EQ.1)I=I-2.**(NBRBIT-1)
       RETURN
       END
       ******
       * LISTING OF THE BASIC VERSION *
       ************************
       *******
       * MAIN PROGRAM BASIC *
       *******
010
       DIM 8(36)
020
       DIM 8(36)
030
       LET N0=16
```

FOR J=1 TO NØ 040 LET B(J) = -1050 060 NEXT J PRINT " I =" 070 INPUT I 080 PRINT " I1=" 090 INPUT I1 100 110 IF I1=0 THEN GOTO 190 120 FOR J=1 TO NØ 130 PRINT " BIT"; J; " =" INPUT K 140 150 IF K>1 OR K<0 THEN GOTO 170 160 LET B(J) = K170 NEXT J REM " CALL SUBROUTINE BIT01 180190 GOSUB 1000 200 PRINT " SORTIE DES RESULTATS" PRINT " I = "; I210 220 PRINT " ETAT DES BITS DE I" FOR J=1 TO NØ 230 240 PRINT A(J) 250 NEXT J 260 STOP \*\*\*\*\* \* SUBROUTINE BASIC \* 1000 REM "....." REM " SUBROUTINE BIT01 1010 REM " ARGUMENTS : I, I1, A(), B(), N0" 1020 REM "....." 1030 1040 LET N1=N0-1 1050 LET I2=I 1060 LET A(N0)=0 1070 IF I2>=0 THEN GOTO 1100 1080 LET A(N0)=1 1090 LET 12=12+2\*\*(NØ-1) 1100 FOR J=1 TO N1 LET I3=INT (12/2) 1110LET A(J)=12-13\*2 1120 1130 LET I2=I3 NEXT J 1140 1150 IF I1=0 THEN RETURN 1160 FOR J=1 TO NØ IF B(J)<>0 THEN GOTO 1210 1170 1180 LET A(J) = 01190 LET B(J) = -11200 GOTO 1240

1210	IF B(J)<>1 THEN GOTO 1240
1220	LET A(J)= 1
1230	LET B(J)=-1
1240	NEXT J
1250	LET I=A(N1)
1260	LET N2=N1-1
1270	FOR J=N2 TO 1 STEP -1
1280	LET I=I*2+A(J)
1290	NEXT J
1300	IF A(N0)<>1 THEN RETURN
1310	LET I=I-2**(N0-1)

1320 RETURN

CONCLUSION : \*\*\*\*\*\*\*

THE USER CAN DIRECTLY VERIFY OR HANDLE BITS' LEVEL OF A MEMORY NORD WHITOUT A SPECIFIC ASSEMBLER OR MACHINE CODE WHICH ARE PARTICULAR TO A GIVEN COMPUTER; THIS SUBROUTINE IS COMPLETELY TRANSPORTABLE TO ANY TYPE OF MACHINE.

THE READER WILL NOTICE SOME DIFFERENCES BETWEEN THE TWO HIGH LEVEL LANGUAGES AND ESPECIALLY WILL APPRECIATE THE MNEMONIC AND RELATIVE STATEMENTS QUALITIES OF FORTRAN.

I recently came across a problem with the RT-11 V5 Extended Memory monitor on 18-bit Qbus systems. I would like to share this problem (and its solution) with you and the other users out there.

The problem, simply (??) stated, is that DEC has been supplying the LSI 11/23+ chip set for some time, and thus some LSI systems (ours, for example) already have 22-bit addressing capability, even though the CPU is plugged into an 18-bit Qbus. Under these conditions, the memory sizing routine in RT-11 V5 does a "wrap-around", that is, the upper four bits of a 22-bit address are ignored, making the next address after "777777" equal to "000000" instead of "1000000". This is not especially critical in the Single Job monitor, where it simply causes the RESORC routine to report 4Mb of memory, and the VM Virtual Memory Disk Emulator thinks the same thing. Using the VM driver under these conditions can cause RMON to be written over and will generally cause the system to crash.

In the RT-11 V5 Extended Memory monitor, the boostrap routine sizes memory and reports to RT-11 that it has 4Mb available. This causes the XM monitor to crash on loading, so users can't even have the use of the background partition.

Since Plessey Peripheral Systems' main product line is Qbus systems, this state of affairs was wholly unacceptable, and some sort of "work around" or patch had to be developed. The result of our work consists of two unsupported patches, either or both of which may be installed to fix this problem. Since these patches alter the RT-11 source files, care must be exercised in using them; i.e., be sure you have adequate backups of your distribution. These patches are presented as a courtesy only. We have tested these patches on the DEC RT-11 distribution and have found them to work as described. However, Plessey Peripheral Systems makes no guarantee as to the accuracy or functionality of these patches, and will in no case provide support for systems on which they are applied. Plessey will assume no responsibility for any damages resulting from the use of these patches.

The first patch is an addition to SYSGEN of a new parameter, MODE22. This parameter turns on/off 22-bit addressing in the RT-11 system, both in

the extended memory monitor and the VM driver. The patch consists of three SLP files, to be applied to SYSGEN.COM, BSTRAP.MAC, and VM.MAC, respectively.

The first SLP file, SYSGEN.SLP, is as follows:

-/	.IFF <es< th=""><th>CAPE&gt; .GO</th><th>TO MO20/,.</th><th></th><th></th><th></th></es<>	CAPE> .GO	TO MO20/,.			
	.IFF KES	CAPE> .GO	TO Q3A			
-/	.GOTO Q3.	, ,				
.Q3A:	.IFF XM					
	-	Transfer (17) and the second	and a second sec	i want 22.	-bit support	(Y)?
-/			OSUB LSC/			
	and the second second		OSUB VM22			
-/.CTLP9	O:.RETURN	/				
.VM22:	.IFT XM	RETURN				
	;				1.74	(*)0
	_	NUE>1 MOD	ESS DO AOI	i want 22	-bit support	(1)?
1 05 . 1	. RETURN					
-/.G5:/	CIERC AD	T MHODEOO	22 hit	annout #		
	.GOSUB S		,22-bit su	ipport.		
/	.00305 3	21				
/						
The second	SLP file,	BSTRAP.S	LP, is as	follows:		
-/.SBTTI	. =	Extended	Memory B	ootstrap	<b>#</b> /	
	MODE22					
	MODE22					
	BIS			BIT	#20,@#SR3/	
	F MODE22					
		#MODE22,				
.IIF EQ	PDT\$OP	NOP				

1

BCS

BIT

It should be noted that the first line of this file should be read as "minus slash period SETTL tab asterisk eight spaces Extended Memory Bootstrap nine spaces asterisk slash"

The third SLP file, VM.SLP, is as follows:

20\$

#MODE22, @#SR3

-/MODE22 = 000020	/,.		
.IIF NDF MODE22	MODE22	=	000000
.IIF NE MODE22	MODE22	=	000020
1			

Once these files have been created, they may be implemented by the following commands:

#### .R\_SLP \*SYSGEN.COM=SYSGEN.COM,SYSGEN.SLP \*BSTRAP.MAC=BSTRAP.MAC,BSTRAP.SLP \*VM.MAC=VM.MAC,VM.SLP \*C

Having implemented these files, perform a SYSGEN, or edit your SYSGEN.CND file to include a line:

#### NODE22 = 000000 ;22-bit support

which will disable 22-bit support in both the Extended Memory monitor and in the VH driver. When you perform a SYSGEN, you will notice a new question:

#### Do you want 22-bit support (Y)?

This question will be asked if you select the XM monitor, or, if you don't select the XM monitor, when you select the VM Virtual Memory Driver. Please note that the default base address of the VM driver in XM systems (BASE=10000) will make it uninstallable on 18-bit systems.

The second patch consists of the addition of a SET command to the VM driver. The patch is implemented through an SLP file, **VMSET.SLP**, the text of which is as follows:

-/REINST	:/,/.EVEN	/
<b>REINST:</b>	.ASCIZ	"?VM-W-Revome/install VM"
	.EVEN	
	.IF EQ M	NG\$T
BAREA:	.BYTE	17,10
	. BLKW	
	.BLKW	
	.WORD	256.
	.WORD	0
C.BT22:	MOV	R3,V.BIT
	MOV	R3,I.BIT
	BR	PRI
	. ENDC	
-/.DRSET	1,.	
	.IF EQ M	MG\$T
	.DRSET	22BIT, 1, S. BT22, NO
	. ENDC	
	.DRSET	
-	.IF EQ M	
S.BT22:		(PC)+,R3
	.WORD	21
	DEC	R3
	BR	C.BT22
	.ENDC	
	.ENABL	LSB
-/.ENDC	/./\$\$.SET	/
	.ENDC	-
PRI:	MOV	PC.RO
	100	•
	ADD	#REINST,RO

5\$:			
10\$:	RTS	PC	
	.DSABL	LSB	
	\$\$.SET	=.	
-/BIS	#HODE22,	e#114SR3/,/BEQ	20\$/
	MOV	(PC)+,R1	
I.BIT:	.WORD	MODE22	
	MOV	#MMSR3,R2	
	BIS	R1, @R2	
	BCS	20\$	
	BIT	R1, @R2	
	BEQ	20\$	
-/CLR	e#MisR3/	,/JMP 100\$	/
	CLR	€R2	
	BR	100\$	
-/.WORD	MODE22/,		
V.BIT:	.WORD	MODE22	
/			

This patch is implemented through the following commands:

#### .R\_SLP \*VM.MAC=VM.MAC,VMSET.SLP \*^C

After entering these commands, either recompile the VM driver or perform a SYSGEN. Upon completion, a command of the form:

#### SET VM [NO]22BIT

will be available.

Please note that these two patches are in no way incompatible; that is, they may both be in place at once. Note as well that the second patch, which implements a SET command for the VM driver only, does nothing for the Extended Memory Monitor problem. It should also be noted that, while the second patch in no way alters the functionality of the VM driver, it does change the warning message printed after a SET command from:

#### ?VM-W-Remove and reinstall this handler

to:

#### ?VM-H-Remove/install VM

While this is not a big deal for most users, this message might be confusing to less sophisticated users, since it can't be found in any of the manuals.

Please note that neither of these patches will correct the RESORC report that there are 4Mb of memory installed. It will, however, fix any problems connected with using the VM driver or the Extended Memory monitor. Please note as well that any software that directly manipulates the memory management registers of the 11/23 will need to be written to take the problems of the 18-bit bus into account.

I hope that these patches will be of use to those DEC and DEC compatible users who have been a little perplexed at some of the glitches in RT-11 V5's extended memory features.

Russell L. Morrison II Plessey Peripheral Systems P.O. Box 19616 Irvine, CA 92714

(714)540-9945

Sincerely,

The Cursor Driven Command File Driver does the following:

1. You give the program the name of a menu file which is displayed on the screen. A menu file has the name of previously created command files together with a short description.

2. You move the cursor anywhere within the command file name. This name must be alphanumeric and can be 1 to 6 characters.

3. You hit the return key. If the command file is created properly and is spelled correctly on the menu, it will then execute.

A couple of comments/observations on the program:

1. It runs on a LSI 11/23 under TSX+; a VT100 terminal in ANSI mode.

2. The escape sequences which we use as a standard at the top of EVERY menu are as follows; (I tried the program an out of date..kind of off the wall sequence and it didn't work..so be warned):

ESC = the escape char. cr = Car. return; lf = line feed.

ESC(ESC[?31ESC[2JESC1;24rESC[HESC[Omerlf

ESC( = If in VT52 mode reset to ANSI

ESCE?31 = If screen 132 col set to 80; the question mark is not part of the sequence but since we have C-ITOH's that use it as part of their sequence we put it in and it works on a VI100 ok.

ESC[2] = Erase entire screen.

ESC[1;24r = Set top-bottom scrolling region.

ESCIH = Cursor unconditionally to HOME position.

ESCCOm = Clear all attributes.

3. You may move the cursor anywhere with the six character command file name and the program will work. If you put a call to this program and the menu names in all command files referenced by your menu system, it can make getting around the system, significantly faster.

I'd like to thank Bruce Johnson of ITI for showing me much faster/easier ways out of the trenches at various and sundry times.

Hope you have fun with it; Any comments, experiences in implementing it, etc. would be most welcome.

I created it out of sheer frustration (ain't it always the way?), after seeing DEC350.

RONALD ROSENTHAL HQ CECOM U.S. ARMY ELECTRONICS COMMAND DRSEL-MS-0 FT. MONMOUTH, NJ 07703 (201) 544-2109

```
(*$L+,$A+*)
PROGRAM PAS;
1
                                                                1
PROGRAM LOGIC;
                                                                1
       READ MENU INTO ARRAY
                                                                1
       MOVE CURSOR TIL CR
                                                                1
       GET CURSOR POSITION
                                                                1
       STUFF INTO X, Y
                                                                1
       GET COMMAND STRING
                                                                1
       SETUP ASSEMBLER LINK TO COMMAND STRING INTERPRETER
                                                                1
       EXECUTE COMMAND STRING AND
                                                                1
       EXIT PROGRAM
                                                                V
CONST
   MAXCOL=115;
   MAXROW=26;
       MINCOL=1;
       BLANK=040B;
       ESCAPE=33B;
       LEFTBR=133B;
       SIX=066B;
       SMALLN=156B;
        UPARROW=136B;
        SURPRISE=041B;
        CURSORRIGHT=103B;
         CURSORLEFT=104B;
         CURSORUP=101B;
         CURSORDOWN=102B;
              CR = 15B;
              LF = 12B;
       NINE=9;
 TYPE
   R = 1. MAXROW;
   C = 1..MAXCOL;
   MN = ARRAY ER, CJ OF CHAR;
   NAME = ARRAY [C] OF CHAR;
   CURPOS = FILE OF NAME;
       CHARSET=SET OF CHAR;
  VAR
       GOOD: BOOLEAN;
       NAMEMENU: ARRAY [1..14] OF CHAR; [MENU FILE NAME FROM COMMAND FILE]
       F: FILE OF CHAR; [ MENU FILE POINTER ]
       D: CHAR; [ RECEIVES CHARS FROM F]
       MENU: MN; [ STORES SCREENMENU IN 2D ARRAY. ]
       BUFFER: NAME; I TEMP STORAGE FOR NUMBERS FROM <READ CURSOR POSITION> ]
```

ENDOFESCAPESEQUENCE: INTEGER; SCREEN: TEXT; ROW: R; MENUCOL, SCREENCOL, COL: C; INCREMENT, LEFTCOL, RIGHTCOL: INTEGER; LENGTHOFARRAY, LENGTHOFSTRING, DONE, COLBOUNDARY, DPOS: INTEGER; STOPCHAR, S, SAMPLE: CHAR; JOBSTAT ORIGIN 44B: INTEGER; CMDLENGTH ORIGIN 510B: INTEGER; CMDFLE ORIGIN 512B: ARRAY [1.. 9] OF CHAR; . PROCEDURE PO20BEEP; CONST DING =  $7B_i$ DONG = 7B;BEGIN WRITE (CHR(DING), CHR(DONG)); END; [PO20BEEP] /\*<----->\*/ PROCEDURE POBOEXECUTECOMMANDFILE; /\* THIS PORDCEDURE CALLS AN ASSEMBLY LANGUAGE MACRD CALL\*/ /\* TO EXECUTE THE STRING OF CHARACTERS ALREADY BUILT. \*/ BEGIN /\* JOBSTAT SETS A BIT IN THE JSW INDICATING THERE'S A COMMAND FILE\*/ /\* TO BE EXECUTED WHEN THE EXIT MACRO IS EXECUTED. \*/ JOBSTAT := JOBSTAT + 4000B;E\*\$C . MCALL . EXIT CLR RO . EXIT \*1 END; [END PROCEDURE] /\*(---->\*/ PROCEDURE PO40CREATECOMMANDFILE; /\*VAR INTERNAL\*/ VAR CMDCOL, I1: INTEGER; BEGIN /\* LENGTH OF COMMAND STRING SHOULD BRE SET HERE; IT WILL BE PASSED TO THE JOBSTATUS AREA WHEN COMMAND FILE IS EXECUTED\*/ CMDLENGTH := NINE; FOR I1 := 1 TO 9 DO BEGIN CMDFLE [11] := ' '; END; CMDCOL := 2;CMDFLE [1] := ' '; CMDFLE [8] := CHR(CR); CMDFLE [9] := CHR(LF);

FOR I1 := LEFTCOL TO RIGHTCOL DO BEGIN CMDFLE [CMDCOL] := MENU [ROW, I1]; CMDCOL := CMDCOL + 1;END; END; [END PROCEDURE] PROCEDURE POSOTRUEFALSE (VAR CH: CHAR; SKIPSET: CHARSET; VAR DB: BODLEAN); /\*VAR INTERNAL\*/ BEGIN IF CH IN SKIPSET THEN BEGIN DB:=TRUE; END; END; [END OF POSOTRUEFALSE] PROCEDURE POSOTESTCHAR; /\*VAR INTERNAL\*/ BEGIN GOOD: =FALSE; IF (SAMPLE >= 'A') AND (SAMPLE <= 'Z') THEN BEGIN GOOD: =TRUE; END; POSOTRUEFALSE (SAMPLE, ['A'...'Z'], GOOD); PO5OTRUEFALSE (SAMPLE, ['0'...'9'], GOOD); END; [END PROCEDURE] /\*<-----PROCEDURE PO7OFINDMENUCOL (DMENU: MN; DROW: R; SCREENCOL: C; VAR ACTUALMENUCOL: C; VAR ENDE G SCAPE: INTEGER); /\* THIS ROUTINE IS TO TAKE THE COLUMN NUMBER RETURNED BY THE CURSOR POSITION\*/ /\* REPORT AND CORRELATE IT TO THE ACTUAL COLUMN POSITION IN THE ARRAY OF THE MENU K GEPT IN CORE. #1 /\*THE CPR DID NOT COUNT ESCAPE SEQUENCES AND WHEN A TAB WAS ENCOUNTERED IT\*/ /\* ACTUALLY INSERTED TABCOUNT (USUALLY EIGHT) NUMBER OF SPACES IN THE COLUMN\*/ /\* NUMBER; WHEREAS IN THE CORE ARRAY THERE IS ONLY 1 CHARACTER (11B ELEVEIN\*/ /\* OCTAL). \*/ CONST TAB=11B; TABCNT=8; VAR DONE, APPARENTMENUCOL: INTEGER; CH: CHAR; BEGIN DONE: =0;ACTUALMENUCOL: =0; APPARENTMENUCOL: =0; REPEAT BEGIN

ACTUALMENUCOL := ACTUALMENUCOL + 1; CH := DMENU [DROW, ACTUALMENUCOL]; IF CH = CHR(ESCAPE) THEN

CH = CHR(ESCAPE) THEN

ACTUALMENUCOL := ACTUALMENUCOL + 1; CH := DMENU EDROW, ACTUALMENUCOL ;

IF CH = CHR(LEFTBR)

BEGIN ENDESCAPE := ACTUALMENUCOL + 2; ACTUALMENUCOL := ACTUALMENUCOL + 2; END

ELSE

THEN

BEGIN ENDESCAPE := ACTUALMENUCOL + 1; ACTUALMENUCOL := ACTUALMENUCOL + 1; END;

ACTUALMENUCOL := ACTUALMENUCOL + 1;

END;

BEGIN

/\* THIS VARIABLE IS TO STOP THE LEFT SCAN OF P110FINDSTRINGBOUNDARY FROM\*/ /\* OVERSHODTING ITS TARGET WHEN SCANNING LEFT; (IT IS ASSUMED THAT DNLY THE\*/ /\* HIGHLIGHT SEQUENCE <ESCIIM> STOP SEQUENCE <ESCIIM>; OR DOUBLE HEIGHT DOUBLE\*/ /\* WIDTH <ESC#N> WILL BE USED.) \*/

/\*THIS COULD HAPPEN IF COMMAND FILE NAME IS RIGHT AGAINST\*/

/\* THE ESCAPE SEQUENCE FOR HIGHLIGHTING ON THE MENU; E.G., ESCLIMXXXXXX N'EST PAS?? \*/

```
IF CH = CHR(TAB) THEN
        BEGIN
                ACTUALMENUCOL := ACTUALMENUCOL + 1;
                 APPARENTMENUCOL := APPARENTMENUCOL + TABCNT;
        END;
IF
    CH = CHR(CR) THEN
        BEGIN
                 DONE: =1;
        END;
IF CH > CHR(37B) THEN
        BEGIN
                 APPARENTMENUCOL := APPARENTMENUCOL + 1;
        END;
IF APPARENTMENUCOL = SCREENCOL THEN
        BEGIN
                 DONE: = 1;
```

END;

END UNTIL DONE = 1;

END; [P070FINDMENUCOL]

/\*<----->\*/ PROCEDURE POBOTESTFORESCAPEBOUNDARY (DCOL, ENDCOL: INTEGER; VAR DB: BOOLEAN); BEGIN IF DCOL = ENDCOLTHEN BEGIN DB := FALSE; END; END; [POSOTESTFORESCAPEBOUNDARY] /\*<----->\*/ PROCEDURE PO90TESTFORSTRINGLENGTH (COLNOW, COLBEGIN, LS: INTEGER; VAR DB: BOOLEAN); /\*VAR INTERNAL\*/ BEGIN IF ABS (COLNOW - COLBEGIN) > LS THEN BEGIN DB := FALSE; END; END; LEND PROCEDUREJ /\*<---->\*/ PROCEDURE PIOOTESTCOLIMITS (DCOL, MIN, MAX : INTEGER; VAR DB: BOOLEAN); /\*VAR INTERNAL\*/ BEGIN IF (DCOL < MIN) OR (DCOL > MAX) THEN BEGIN DB := FALSE;END; END; LEND PROCEDURE] / St. C were made and and and a start and a start and a start and and a start and a start and a start and and a start and a st ---->\*/ PROCEDURE P110FINDSTRINGBOUNDARY (ENDESCAPE: INTEGER; DMENU: MN; DROW: R; DCOL: C; S VAR DCOLBOUNDARY: C: VAR INC: INTEGER); /\*VAR INTERNAL\*/ VAR NUMCHARS, DLENGTHOFSTRING : INTEGER; BEGIN NUMCHARS := 0;DLENGTHOFSTRING := 6; COLBOUNDARY: =DCOL; DONE := 0;REPEAT GOOD := TRUE; SAMPLE := DMENU [DROW, COLBOUNDARY]; PO60TESTCHAR; IF GOOD = TRUE THEN BEGIN POBOTESTFORESCAPEBOUNDARY (COLBOUNDARY, ENDESCAPE, GOOD); END; IF GOOD = TRUE THEN BEGIN PO90TESTFORSTRINGLENGTH (COLBOUNDARY, DCOL, DLENGTHOFSTRING, GOOD); END;

IF GOOD = TRUE THEN BEGIN P100TESTCOLIMITS (COLBOUNDARY, MINCOL, MAXCOL, GOOD); END; IF GOOD = TRUE THENBEGIN COLBOUNDARY: = COLBOUNDARY + INC; NUMCHARS: =NUMCHARS + 1; END ELSE BEGIN DONE := 1;COLBOUNDARY := COLBOUNDARY - INC; END; UNTIL DONE = 1;/\* IF SOMEONE PUTS THE CURSOR ON A NO-NO WE WANT TO MOVE IT BACK TO ITS \*/ /\* STARTING POINT. \*/ IF NUMCHARS = 0 THEN BEGIN COLBOUNDARY: = COLBOUNDARY + INC; END; DCOLBOUNDARY := COLBOUNDARY; END; [END PROCEDURE] PROCEDURE P120GETCHAR; /\*VAR INTERNAL\*/ /\*THIS PROCEDURE IS FOR DIPLSAY THE MENU\*/ BEGIN GET(F); D: =F^; END; [P120GETCHAR] PROCEDURE P130DISPLAYSTOREFILE; /\*VAR INTERNAL\*/ BEGIN READ (NAMEMENU); RESET (F, NAMEMENU); COL :=1; ROW :=1; WHILE NOT EDF(F) DO BEGIN /\* DISPLAY MENU AND STORE IN ARRAY IN CORE; DISPLAY SCREEN \*/ D: =F^; IF D=CHR(OOOB) THEN BEGIN END ELSE BEGIN WRITE(D); MENUEROW, COL ]: =D; COL: =COL+1;

END; IF D = CHR(LF) THEN BEGIN COL: =1; ROW: =ROW+1; END; 1\* ONE READ ONE WRITE PER MODULE!!!!!\*/ P120GETCHAR; END; [END OF WHILE] CLOSE(F); END; [P130DISPLAYSTOREFILE] PROCEDURE P140ECHOFF; /\*VAR INTERNAL\*/ BEGIN WRITE(CHR(035B), 'F'); END; [P140ECHOFF] PROCEDURE P150ECHON; /\*VAR INTERNAL\*/ BEGIN WRITE(CHR(035B), 'E'); END; [P150ECHON] /\*<-----PROCEDURE P160GETCURSORPOSITION; BEGIN WRITE(CHR(ESCAPE)); WRITE(CHR(LEFTBR)); WRITE(CHR(066B)); WRITE(CHR(156B)); END; [P160GETCURSORPOSITION] / 🔆 🤇 man and also and the loss and PROCEDURE P170GETCURSORPOSITION; BEGIN P160GETCURSORPOSITION; END; [P170GETCURSORPOSITION] /\*<-----\_\_\_\_\_ PROCEDURE P180GETSCREENCHAR (VAR CH: CHAR); /\* THIS PROCEDURE WILL BE MADE AN EXTERNAL PROCEDURE THAT\*/ /\* WILL BE CALLABLE FROM ANY PASCAL PROGRAM. \*/ VAR JOBSTAT ORIGIN 44B: INTEGER; /\* B & C ARE ON AND DFF VT100 ESCAPE LETTER ACTIVATION. \*/ /\* S & T ARE ON AND DFF SINGLE CHAR. ACTIVATION \*/ /\*.WHEN ON YOU DON'T NEED A CR TO RETURN A CHAR TO YOU \*/

/\* JOBSTAT DOES THE SAME THING FOR RT11 AS S DOES FOR TSX+ \*/ /\* SEE PROCEDURE PO85. FOR TURNING OPTIONS OFF. \*/ BEGIN WRITE (CHR(035B), 'B'); WRITE (CHR(035B), 'S'); JOBSTAT := JOBSTAT OR 10000B; REPEAT C\$C . MCALL . TTYIN . TTYIN MOVB RO, @CH(6) UNTIL CH # CHR(O); END; [P180GETSCREENCHAR] PROCEDURE P190REVERSEP1800PT10NS; /\* THIS PROCEDURE WILL BE MADE AN EXTERNAL PROCEDURE THAT\*/ /\* WILL BE CALLABLE FROM ANY PASCAL PROGRAM. \*/ VAR JOBSTAT ORIGIN 44B: INTEGER; /\* B & C ARE ON AND DFF VT100 ESCAPE LETTER ACTIVATION. \*/ /\* S & T ARE ON AND OFF SINGLE CHAR. ACTIVATION \*/ /\* WHEN ON YOU DON'T NEED A CR TO RETURN A CHAR TO YOU \*/ /\* JOBSTAT DOES THE SAME THING FOR RT11 AS S DOES FOR TSX+ \*/ /\* SEE PROCEDURE POBO. . FOR TURNING OPTIONS ON. \*/ BEGIN WRITE (CHR(035B), 'C'); WRITE (CHR(035B), 'T'); JOBSTAT := JOBSTAT AND NOT 10000B; END; LEND PROCEDURE P190REVERSEP1800PTIONS] \_\_\_\_\_\*/ PROCEDURE P200SCANTIL; VAR X: INTEGER; BEGIN FOR X: = 1 TO LENGTHOFARRAY DO BEGIN BUFFERCXJ := ' '; END; X:=0; WHILE S # STOPCHAR DO BEGIN X:=X+1; BUFFERCX1 := S; P1BOGETSCREENCHAR(S); END; LENGTHOFSTRING := X;

END; [P200SCANTIL]

```
PROCEDURE P210ARRAYTONUM;
/*VAR INTERNAL*/
VAR I1, I2: INTEGER; R1: REAL;
BEGIN
R1:=0.0;
12 := 0;
DPDS:=0;
FOR I1: =LENGTHOFSTRING DOWNTO 1 DO
       BEGIN
              R1 := R1+((ORD(BUFFER[I1]) - ORD('O')) * EXP10(I2));
              12 := 12 + 1;
       END;
DPOS := TRUNC(R1);
END; [P210ARRAYTONUM]
/*<---->*/
PROCEDURE P220MOVECURSOR;
VAR ENDSW: INTEGER;
BEGIN
ENDSW := O;
REPEAT
              BEGIN
P180GETSCREENCHAR(S);
IF S = CHR(ESCAPE) THEN BEGIN
                             P1BOGETSCREENCHAR(S);
                       END;
IF S = CHR(LEFTBR) THEN BEGIN
                             P1BOGETSCREENCHAR(S);
                       END;
IF S = CHR(CURSORRIGHT) THEN
                             BEGIN
                             WRITE (CHR(ESCAPE));
                             WRITE (CHR(LEFTBR));
                             WRITE (CHR(103B));
                             END;
IF S = CHR(CURSORLEFT) THEN BEGIN
                             WRITE (CHR(ESCAPE));
                             WRITE (CHR(LEFTBR));
                             WRITE (CHR(CURSORLEFT));
                       END;
IF S = CHR(CURSORUP) THEN BEGIN
                            WRITE (CHR(ESCAPE));
                            WRITE (CHR(LEFTBR));
                            WRITE (CHR(CURSDRUP));
                      END;
IF S = CHR(CURSORDOWN) THEN BEGIN
                             WRITE (CHR(ESCAPE));
                             WRITE (CHR(LEFTBR));
                             WRITE (CHR(CURSORDOWN));
                      END;
/*IF S = CHR(CR) THEN
                     BEGIN
                             ENDSW := 1;
                     END;
```

\*/ IF S = CHR(LF) THEN

BEGIN ENDSW := 1; END;

END LEND OF REPEAT

UNTIL ENDSW = 1; END; [P220MOVECURSOR]

PROCEDURE P230STUFFINTOXY; /\*VAR INTERNAL\*/ BEGIN STOPCHAR := '; '; LENGTHOFARRAY := 2; P180GETSCREENCHAR(S); P180GETSCREENCHAR(S); P180GETSCREENCHAR(S); P200SCANTIL; P210ARRAYTONUM; ROW := DPOS; P1BOGETSCREENCHAR(S); STOPCHAR := 'R'; P200SCANTIL; P210ARRAYTONUM; SCREENCOL := DPOS; END; [P230STUFFINTDXY]

#### 

/\*MAIN PROGRAM\*/

BEGIN P130DISPLAYSTOREFILE; P220MOVECURSOR; P160GETCURSORPOSITION; P230STUFFINTOXY; /\* THESE NEXT TWO STATEMENTS ARE FOR DEBUGGING PURPOSES COMMENTED OUT \*/ /\*WRITE ('SCREEN', SCREENCOL); \*/ /\*WRITE ('ROWXXX', ROW); \*/ PO7OFINDMENUCOL (MENU, ROW, SCREENCOL, MENUCOL, ENDOFESCAPESEQUENCE); INCREMENT: =-1; P110FINDSTRINGBOUNDARY (ENDOFESCAPESEQUENCE, MENU, ROW, MENUCOL, LEFTCOL, INCREMENT); INCREMENT: =1; P110FINDSTRINGBOUNDARY (ENDOFESCAPESEQUENCE, MENU, ROW, LEFTCOL, RIGHTCOL, INCREMENT); P040CREATECOMMANDFILE; /\* SAME METHOD USED HERE FOR THIS DEBUGGING STATEMENT \*/ /\*WRITE (LEFTCOL, RIGHTCOL, CMDFLE);\*/ P190REVERSEP1800PTIONS; POBOEXECUTECOMMANDFILE; END.

#### USER REQUESTS

I am running RT-11 on a PDP 11/23 to prepare a TU-58 to run stand-alone on a 11/04. What do I have to change in FRT.MAC which is part of the stand-alone module? Where can I get the latest documentation on FRT.MAC and SIMRT.MAC. Joseph F. Heinig NASA Goddard Space Flight Center Code 564.3 Advanced Systems Section Greenbelt Road Greenbelt, Md. 20771

We are in urgent need of a <u>serial</u> handler for a printer with

X-on/X-off protocol for operation under Version 3 of RT-11.

If you can advise us as to where we might find such a handler, we would be most grateful.

Very truly yours,

VARTRON CORPORATION

750 WELCH ROAD PALO ALTO, CALIFORNIA 94304 PHONE: (415) 328-2531

aulan Pat Vartanian

DECUS Associate 118501

## USER RESPONSES

The very usefull programm DATE, published by R.M.Harrington in Minitasker March 1983 Vol 9, No.1, could be added with some lines to accept time from 20: to 23: hours (for night-workers!)

Change the lines between the comment "HH OR H FORMAT" and "NOW CHECK FOR ERRORS" as follows:

;	HH	OR	Η	FORMAT	
;					
HH:		CMP	)	R1,并541	
		BPI	1	TIM	; too much input
		MOV	7	#34460, R2	; Range 0-9
		JSR	2	PC, CHECKR	
		CMP	B	-l (Rl),#40	; CK for Space
		BEC	)	CKE	

	CMPB	-l (Rl),#61	; CK for l
	BEQ	ADJ	
	CMPB	-l (Rl),#62	; CK for 2
	BNE	TIM	
	INC	RL	
	MOV	<b>#</b> 31460, R2	; Range 0-3
	JSR	PC, CHECKR	
ADJ:	DEC	RL	; Adjust Pointer
	CMPB	-1 (R1),#40	; CK for Space
	BNE	TIM	
CKE:	CMPB	-2 (Rl),#'E	; CK for 'E' of time
	BNE	TIM	
;			

; NOW CHECK FOR ERRORS

L.Kahlbau c/o SIEMENS,SARL Fertigungstechnik Estr.de Almeirim 7000 Evora PORTUGAL

Yours sincerely

150

L. Kahlbau

UPCOMING SYMPOSIUM INFORMATION

ANNOUNC I NG

RT-11 SESSION NOTES

# DECUS LAS VEGAS

There will be a volume of Session Notes containing the visuals for some of the RT-11 papers to be given at the DECUS 1983 Fall Symposium in Las Vegas. Additionally, the volume will contain "The Best of RT-11, Volume 2", as an added bonus.

Look for the document at the DECUS store at the Symposium.

#### 1983 FALL SYMPOSIA IN LAS VEGAS

Even though there was very little time between the Spring symposia and the Fall symposia, I received a record number of submissions for the RT-11 SIG. The scheduling problem was compounded by the fact that the number of meeting rooms was less than before. The end result of all this is a schedule that is a little different than before. First of all, we will be starting at 8:30 in the morning instead of 9:00. Second, the coffee break time was eliminated so that we could hold more sessions. Cookies and milk will be available in a number of locations so that you will not have to go the entire morning or afternoon without food. Finally, you will notice that a number of RT-11 sessions are scheduled for Friday morning and afternoon. The thought here was that this was really a five-day convention and that we should better utilize Friday in order to avoid "session burnout." As always, I will be available at the symposia to listen to constructive criticism only so long as you buy the beer.

I have attached a Master Index of all the RT-ll sessions to enable you to make plans to attend the symposia. I hope to see you there.

RT-11

CODE	TITLE/SPEAKER	TIME REQ.
R001	USING A PDP-11/23 AS A FILE SERVER FOR ATTACHED LSI-11'S Fouts, Martin	l hour
R002	MIGRATION OF DBMS SOFTWARE FROM RT-11 TO RSX-11M Natale, Robert C.	30 minutes
R004	RT-11 USERS SPEAKOUT Rhodes, Ned W.	2½ hours
R005	RT-11/TSX-PLUS COMPATIBILITY ISSUES Peterson, Jack J.	l hour
R007	RT-11 SIG BUSINESS MEETING Rasted, John T.	30 minutes
R008	RT-11 SIG SYMPOSIUM WRAP-UP Rasted, John T.	30 minutes
R009	DECUS LIBRARY LAYERED PRODUCTS PANEL FOR RT-11 Bourgeois, Nick	$1^{l}_{2}$ hours
R010	RT-11 ROADMAP Rasted, John T.	30 minutes
R011	RT-11 USER APPLICATION WORKSHOP Rasted, John T.	l hour
R012	RT-11 USER COMMAND LINKAGE Crowell, John M.	30 minutes

R013	KT-11 FUTURES WORKSHOP Crowell, John M.	1 hour
R014	HOW TO DEVELOP RT-11 DEVICE HANDLERS Rhodes, Ned W.	1 hour
R017	COMBATTING FLASH FLOODS WITH PDP-11S Peterson, Jack J.	1 hour
R018	TSX-PLUS INTERNALS Bramlet, Jan	1 hour
R019	ACCESSING MEMORY ABOVE 56KB FROM RT-11 FORTRAN Trellue, Ron	1 hour
R020	SHARED REGIONS AND RESIDENT LIBRARIES FOR RT-11 XM Adams, Greg	1 hour
R021	RT-11 XM NEW USER Adams, Greg	l hour
R022	RT-11 FEEDBACK SESSION RT-11 Software Development Group	l hour
R023	RT-11 LANGUAGES PANEL RT-11 Software Development Group	30 minutes
R023 R024		30 minutes 1 hour
	RT-11 Software Development Group RT-11 PRODUCT PANEL	
R024	RT-11 Software Development Group RT-11 PRODUCT PANEL RT-11 Software Development Group RT-11 DIRECTORY STRUCTURES INTERNALS	1 hour
R024 R025	RT-11 Software Development Group RT-11 PRODUCT PANEL RT-11 Software Development Group RT-11 DIRECTORY STRUCTURES INTERNALS Gentry, Martin RT-11 IND NEW USER	l hour l hour
R024 R025 R027	RT-11 Software Development Group RT-11 PRODUCT PANEL RT-11 Software Development Group RT-11 DIRECTORY STRUCTURES INTERNALS Gentry, Martin RT-11 IND NEW USER Metsch, James USING TSX-PLUS SHARED RUN-TIME SYSTEMS	1 hour 1 hour 1 hour
R024 R025 R027 R029	RT-11 Software Development Group RT-11 PRODUCT PANEL RT-11 Software Development Group RT-11 DIRECTORY STRUCTURES INTERNALS Gentry, Martin RT-11 IND NEW USER Metsch, James USING TSX-PLUS SHARED RUN-TIME SYSTEMS Crapuchettes, Jim IMPROVING PERFORMANCE OF RT-11 FORTRAN PROGRAMS	1 hour 1 hour 1 hour 1 hour

# PAST SYMPOSIUM INFORMATION

From: William K. Walker Monsanto Research Corp. P. O. Box 32 OS-123 Miamisburs, Ohio 45342 (513) 865-3557

I save a short presentation during the Foreign Peripherals Forum at the St. Louis DECUS meeting on the model 306A clock board from Grant Technology Systems. This is a KWV11-C equivalent board that also includes a really slick battery-backed calendar clock option. A number of people expressed interest in a couple of utilities which I had written to set the calendar clock and to set the RT-11 date and time from the clock values. This stuff was not ready for the RT-11 SIG tape at the time and I didn't have any listings with me. I have since found time to clean-up these routines and to add some additional code and assembly conditionals to make them more general. I am enclosing source listings for those of you who might be interested. For those of you who are too lazy to do your own typing, I have also submitted them to the DECUS library.

Sincerely,

Witchie

William K. Walker

.title set306.mac .enabl lc .ident /wkw02/ .nlist cnd

; This program sets up the date and time on the calendar clock option for ; the GTSC model 306A real-time clock/calendar clock board.

; This is NOT a real sophisticated program -- if you tell it to set up ; garbage on the board, it will cheerfully do so. It is, however, simple, ; and relatively easy to understand.

; Note that there are conditionals in the code for three different set-up ; variables. You may choose to read/write the resisters in binary or BCD ; format, you may keep AM/PM or 24-hour clock time, and you can have the ; board compensate for Daylisht Savings Time. Note also that the program ; turns all interrupt enable bits off. The alarm times are undisturbed ; however.

i This program will run under versions 4 and 5 of RT-11 and probably i earlier and later versions as well.

Contributed by: William K. Walker
Monsanto Research Corp.
P. O. Box 32 05-123
Miamisburg, Ohio 45342

.mcall .stlin, .exit ceba = 170400 Base address for clock registers resa = ccba+12Register A address Register B address resb = ccba+13resd = ccba+15Resister D address Conditionals: dm = 4 Disable if board is to operate in BCD mode ck.24 = 2 Disable if board is to keep AM/PM time = 1 Disable if board is not to compensate for daylight savings dse f time .iif ndf dm, = 0 ់ក .iif ndf ck24, ck24 = 0 .iif ndf dse; dse = 0 = 200 set dvrset = 160dvset = 40 bset = set!dm!ck24!dse set306; nov #ccba,r0 Set up to grab current time data off board #sec,r1 ; (really just after alarm times)... mov mov \$6,r2 10\$: @‡resa Update in progress? tstb hmi 10\$ Branch if so 20\$: (r0)+,(r1)+;Get the data... movb r2,20\$ sob @#resd,r0 Set 'valid RAM and time' bit movb .stlin #buf;#yeara Frompt for and set year call aschin Convert to appropriate binary ;Store result movb r0,year .stlin #buf,#montha ;Get, convert, and store month... call ascbin movb r0,month .stlin #buf,#daya ;Do day of month... call ascbin r0,day movb .stlin #buf,#daywka #Do day of week... ascbin call movb r0,daywk .stlin #buf,#houra Hour ... call ascbin movb r0,hour .if ea ck24 .stlin FAM or PM?" #buf,#ampma #240,buf bicb Make response upper case, 7-bit buf,#'A FAM? CMPD 30\$ bea CMPD buf;#'P FPM? bne 30\$ bisb #200,hour #Set FM (hish-order) bit .endc 30\$: .stlin #buf,#mina #Minute... call aschin movb r0,min

	∙stlin call movb	≇buf,≢secœ ascbin r0,sec	;Second
	∙stlin mov mov mov movb movb	<pre>#buf,#seta #sec,r0 #ccba,r1 #10,,r2 #bset,@#resb #dvrset,@#resa</pre>	;"Hit <return> to set clock" ;R0 =&gt; data buffer ;R1 =&gt; clock registers ;R2 = no. of bytes to transfer ;Toggle set-up bits ;Reset divider chain</return>
40\$;	movb sob	(r0)+,(r1)+ r2,40\$	fload registers
	mo∨b bicb ∙exit	‡dvset,@‡re⊴a ‡set,@‡re⊴b	\$Remove divider reset \$Start clock \$Exit to RT−11
asebin	•		
0500111	mov clr tstb bne return	#buf,r1 r0 (r1)+ 10\$	<pre>#R1 =&gt; input buffer #Clear R0; will contain result #Test first character #Continue if not null #Return with zero result otherwise</pre>
10\$:	tstb	(r1)	Fest 2nd character
	bea	20\$	;If null, number is 0-9
	movb	-(r1),r0	}Get 10′s character
	bic	#^C<17>,r0	Strip out ASCII stuff
·if ea	dm		
	.rest	4	€Move left 4 bits into hi⊴h-order nibble
	asl	rO	
	.endr		
•iff	venu.		
* # 1 1	asl	rO	<pre>#Multiply by 10. by doing (n#2)+(n#8)</pre>
	mov	r0,r2	MUICIFIS DE IV: DE GUINE (N#2/1(N#0/+++
	asl	r0	
	as1 as1	r0	
		r2,r0	
	add	r29r0	
•endc	4 - 4	(r1)+	Adjust pointer
	tst	(11)+	HODORC POINCEL
20\$:	movb bic	-(r1),r2 #^C<17>,r2	;Get one's character ;Strip out ASCII stuff
.if ea			
7.41 G.G.	bis	r2,r0	;Set low-order nibble
•iff	DT 3	12710	Joev Iow older hibble
•endc	add	r2,r0	;Add to result for 10's character
+enac	return		
buf:	.blkb	134. ;Buffe	r for ,≤tlin request
sec:	.byte	0 #Data	buffer for clock registers
	.byte	0	
min:	.byte	0	
	.byte	0	
hour:	.byte	0	
	.byte	0	
daywk:	.byte	0	
day:	.byte	0	
month:	.byte	0	
year:	.byte	0	

Prompt messages:

yeara: .ascii / Year (1983=83)? /<200> montha: ,ascii / Month (Jan=1)? /<200> daye: .ascii / Day? /<200> daywka: .ascii /Day of week (Sun=1)? /<200> houra: .ascii / Hour? /<200> amemot .ascii ∕ AM or PM (A or P)? /<200> ming: .ascii / Minute? /<200> seco: + ascii 1 Second? /<200> seta: .ascii /Hit <return> to set clock.../<200> .even +end set306 .title setdt.mac .ident /wkw02/ lc .enabl Inlist end ; This program sets the RT-11 date and time from the GTSC model 306A ; clock board. ; The handiest way to use this program is to run it, in your start-up ; command file. It can, of course, be run at any time you may wish to ; bring the RT-11 date and time into agreement with the the clock on the ; 306A. Fhis program will run under a version 4 or 5 monitor. # Contributed by: William K. Walker Monsanto Research Corp. ŝ P. O. Box 32 ŷ 0S - 123Miamisburg, Ohio 45342 ŝ .mcall .sval, .print, .sdttm, .exit Calendar clock base address = 170400ccha #Register A\* address rega = ccba+12#Address of job status word JSW - 44 userrb = 53 Address of user error byte sysver = 276 sever\$ = 10 **;Offset** of monitor version number Severe error bit in user error byte = 1 ;Disable if you do not have EIS instructions eis Remove semicolon if you have 50Hz line-time clock i1t50hz = 1Remove semicolon if board is set up for AM/PM time = 1 13mPm = 1 Remove semicolon if board is set up in BCD mode ;bcd Disable if you don't want date and time printed on exit datime = 1 ; NOTE: The mainline code in this program doesn't really do very much. If you strip out this mainline code and add the appropriate GLOBL and PSECT stuff, you have a MACRO or FORTRAN-callable â \$ subroutine named GSDTM that will set the RT-11 date and time ŝ

from the 306A clock.

\$

setdt:	dual	tanoa.teucuon	;Get RT−11 version number
	bic	#°C<377>,r0	yoet Kr 11 version number
	cmpb	r0,#4	Test for version 4
	bst	20\$	Branch if V5 or later
	bea	10\$	€Branch if V4
	.print	<b>t</b> wrsver	€Complain if earlier than V4
	bisb		b ;Set severe error bit
	elr	rO	jDo a hard exit
	.exit		
10\$;	mov	#4000,chnbit	Set-up for V4-style chain exit
20\$:	call	ssdtm	Set date and time from 306A
.if df	datime		
	mov	\$1000,sp	Pass DATE and TIME commands to RT-11
	mov		<pre></pre>
	mov	<pre>#cstart,r1</pre>	
	mov	<b>#</b> 510, r2	
	mov	r0,(r2)+	
30\$;	movb	(r1)+,(r2)+	
	sob	r0,30\$	
	bis	chnbit,@ <b>‡</b> jsw	
40\$:	clr	rO	
.endc			
	•exit		
chnbit:	.word	40	
		/?SETDT-F-Wrons	Version Of RT-11/
	.even		
.if df	datime		
csize:	.word	cend-cstart	
cstart:	.asciz	/DATE/	
	,asciz	/TIME/	
cend∶ ∙endc	•even		
∮ *Get-a	and-set"	date and time fo	rom GTSC model 306A clock board.
.iif nd	f bed, n	oon = 12.	
.iif df	bed, n	oon = 22	;(22 = 12, in BCD)
ssdtm:			
	mov	<b>#10.,</b> r3	;Get ready to move 10 bytes
	mov		; starting from the cc base address
	mov		; to the local buffer
10\$;	tstb	@tresa	Clock update in progress?
	bmi	10\$	Branch if so
20\$:	movb	(r4)+,(r5)+	Move a byte to the buffer
	sob	r3,20\$	;Keep soins until done
.if df	awsw		€Code for AM/PM time
	tstb	hr	fls time PM?
	bea	30\$	Branch if not
	movb	hr,r0	;Get hours value
	bic		#Mask for hour value
	add		;Make it a 24-hour time value
	movb	r0,hr	<pre>#Store "corrected" value</pre>
30\$:			
.endc			

+endc

.iif df bed, call bedbin ;Convert register values to binary, if necessary elr 15 Clear R5 movb month,r5 ;Get the month ,if ndf eis .rept Shift left 5 bits... 5 as1 r 5 .endr · iff \$5,r5 Shift left 5 bits, EIS code ash .endc bisb daym,r5 Get the day of the month .if ndf eis Shift left 5 bits... +rest 5 as1 r5.endr .iff ash \$5,r5 ;Shift left 5 bits, EIS ,endc movb ;Get the year year, r4 #72. + + 4 sub **;Offset** it from 1972 bis r4,r5 Stuff it into r5 mov r5,date Save result as date word clr rO Clear RO (will be high-order time) clr Clear R1 (will be low-order time) r 1 movb hr, r1 Get hour and cal1 muld60 ; convert it to minutes past midnight movb min+r5 ;Get minutes, add r5,r1 ; add them in; and cal1 mu1d60 ; convert to seconds movb sec;r5 ;Get seconds add r5,r1 ;Add to set seconds from midnisht... ade rO .if ndf lt50hz muld60 (Convert to ticks (60Hz clock) call .iff (Convert to ticks (50Hz clock) call mu1d50 .endc Store high-order time r0, timehi MOV Store low-order time r1, timelo MOV ;Set the RT-11 date and time •sdttm **‡**area,**≇**date return ; Buffer for clock resisters: .byte sec: 0 ,byte Ő min: .byte 0 ,byte 0 hr: ,byte 0 ,byte 0 ,byte 0 daym: .byte Ö month: .byte 0

43

year:

,byte

0

; Date and time words for .SDITM request: 0 date: .word timehi: .word 0 timelo: .word 0 FEMT argument block for .SDTTM 0,0 areat .word ; This routine multiplies a double-precision (two-word) integer by 60. ; It takes advantage of this special case, and does it as (64\*N)-(4\*N). muld60: .if ndf eis Non-EIS code... mov #2,r4 Multiply R0,R1 by 4... clc 10\$: as1 r1 rol rO sob r4,10\$ MOV r0,r2 #Save result (n\*4)... r1,r3 MOV Now multiply by 16 to set n#64 #4,r4 MOV clc 20\$: asl r1 rol rO r4,20\$ sob ;EIS code... .iff Multiply R0,R1 by 4... ashc #2,r0 r0,r2 Save result... mov MOV r1,r3 ashc #4,r0 Now multiply by 16 to set n\*64 +endc r3, r1 ;Subtract R2,R3 from R0,R1... sub sbc rO sub r2,r0 return .if df 1t50hz For those of you with 50Hz line-time-clocks, this routine multiplies ; a two-word integer in R0,R1 by 50. It does this by treating 50\*n as ; (32+16+2)\*n. mu1d50: .if ndf eis Non-EIS code... clc Multiply RO,R1 by 2... 351 r1 rol rO r0,-(5P) ;Save result (n\*2) MOV mov r1,-(sp) #3, 14 Now multiply by 8 to set n#16... mov clc 10\$: asl r1 rol гÕ r4,10\$ sob mov r0,-(sp) #Save result (n\*16)

44

r1,-(sp)

MOV

	clc asl rol	r1 r0	Now multiply by 2 so that R0,R1 is n#32
.iff	ashc mov mov	#1,r0 r0,-(sp) r1,-(sp)	;EIS code to do same as above ;n#2
	ashc mov mov	#3,r0 r0,-(sp) r1,-(sp)	\$n*16
	ashe	<b>#1</b> ,r0	ŧn#32
•ende	add adc add	(sp)+,r1 r0 (sp)+,r0	;Add n*16 to n*32 to set n*48,,,
	ತದರ ತರ ತರರ	(5p)+,r1 r0 (5p)+,r0	;Add n*2 to n*48 to set n*50
∙endc ∙if df	return bcd		;Return
		soes throush the ) to binary,	list of clock register values and changes
bedbin:			
	mov mov	‡sec,r0 ‡10,,r1	\$RO => re⊴ister value list \$R1 = repeat count
10\$:	movb bic bea ror mov ror	(r0),r2 #~C<360>,r2 20\$ r2 r2,r3 r2	;Get a value ;Mask for 10's digit ;Nothing to do if zero ;Multiply this value by 10
	bic add	r2 r3,r2 (r0),r3 ‡^C<17>,r3 r3,r2 r2,(r0)+	FResult in R2 FGet value asain FMask for one's disit FAdd this to previous result FStore binary value
20\$:		r1,10\$	;store binars value ;Go do another until done ;Then return
•endc			
•end	setdt		

# **RT-11 MACRO/FORTRAN Interactions**

John M. Crowell Los Alamos National Laboratory Los Alamos, NM

Ned W. Rhodes, Session Chairperson E-Systems Falls Church, VA

Reported by Gavin Perry, DECUS Scribe Service

This tutorial covered the mechanism for callins MACRO routines from a FORTRAN program. The material is covered in the FORTRAN Users Guide and the FORTRAN Library Guide. When writing FORTRAN programs one sometimes needs to speed up certain critical parts of the program. These time critical routines will be faster if coded in MACRO routines which can be called from a FORTRAN program using the techniques presented here. These techniques include basic information on FORTRAN conventions for passing arguments to a subroutine. Also included are some of the pitfalls encountered when writing FORTRAN callable routines, with hints on how to get around them.

When a CALL statement is issued from a FORTRAN program, the code generated declares the subroutine name as a global and passes the address of an argument block in R5 to the routine as follows:

.GLOBL SUBRTN MOV #ARG,R5 JSR PC, SUBRTN ARG: 3 ; the number of arsuments X ; the address of parameter X Y ; etc for the rest of the arsuments Z

The MACRO routine can now set the argument addresses through R5. The first word of the argument block contains the number of arguments being passed in the low byte. The high byte, while usually 0, is officially undefined. This permits

the use of certain tricks. (see below) The first word in the argument block is followed by the addresses of each of the arguments in the In the subroutine, these parameters may be accessed by indexing R5 (e.s. 2(R5)). This is safer than altering the value of R5, since other routines may also want to reference the arguments pointed to by R5.

#### Function Calls

A FORTRAN function call returns with the answer in RO. If the answer is Integer\*4 or Literal\*4 the low order result will be in RO and the high order part in R1. For a Real variable, the low order portion of the result is in R1 and the high order portion is in R0. Double precision returns four words with R0 containing the most significant portion of the result and the remaining portions in R1 to R3 (least significant). For a complex number the high order portion of the real number is in R0, the low part in R1, while the imaginary portions of the value will be in R2 and R3. FORTRAN expects to find the results returned to it in this format.

#### GOTCHA's

Missing arguments will have -1 as the address, so be sure to check for addresses of -1 when there is a chance of missing arguments in the call (es CALL (A,B,,X)). If no arguments are passed the first word will contain 0 in the low bute, so you may want to check for that too. FORTRAN doesn't care if you save the resisters RO to R5 and it won't save them for you between calls to your routines, so be sure to save any values that will be needed in other calls to a routine. The stack must also be saved. It is very important that for every push onto the stack there is also a pull from the stack. Four out of seven sotcha's were MIND THE STACK. It doesn't matter how many times it's said, everyone sometime ends up leaving a number on the stack. When this happens, a return to PC soes to never-never land. If the stack contains 0 the program will just exit without even saving good bye. Some rushes onto the stack are not obvious. For example, if CSISFC is called it pushes the number of switches onto the stack even if it is zero. Don't modify FORTRAN constants unless you want 2+2=5. It is not the value that is passed but the address. FORTRAN won't know that the yalue of the constant has been changed. If you are using the floating point instruction set and changing defaults for the

precision or the mode, be sure to save the value of the current flags and pop them when done with the different mode or precision. Don't mung R5 until done with all the arguments or you may grab the wrong value.

FILE I/O

FORTRAN has an OTS work area where it keeps track of what I/O channels are open and various program linkages. Tell FORTRAN if you open or close I/O channels by using the library routines IGETC and IFREE respectively. Don't use CSIGEN since it closes channels 0-8 which are often opened by FORTRAN. Use CSISPC instead, but watch the switch number push on the stack. If FORTRAN doesn't know about the channels you open it may set a channel already open when it trys to open a channel.

#### COMMON BLOCK

The common block statement creates a PSECT which you can use from your MACRÓ routines; Just declare that PSECT in the MACRO. See the FORTRAN Users Guide for the format.

### FORTRAN ERROR TRAPS

An error in FORTRAN causes a TRAP instruction with the argument 200 + the error number. You can use these errors to tell the user about fatal errors; be sure to do something that will allow for a graceful exit anyway (such as MOV -1, RO) since a CALL SETERR may have been executed which will keep it from exiting until the error count reaches the count level specified. If there is any chance that error traps will be called (either you call them or you use OTS routines that may trap) be sure that the trap vector has been initialized.

The PSECT layout of a FORTRAN program was discussed. The first statement of a FORTRAN program starts with a call to \$\$0TI to initialize the OTS followed by a pointer to the MAIN which then points to the data to be initialized. It is possible to write threaded code to be used with threaded (OTS) routines and programs but it was not recommended. The threaded structure is a list of entry points to the threaded routines, followed by addresses for the parameters and constants. A naming convention identifies the FORTRAN operation codes the data types and address codes (to tell how many levels of pointers back to go

before you'll reach the value). More information on this is available in the documentation of FORTRAN or from John Crowell.

### MACDBG/RT-11: A User's Critique

John M. Crowell Los Alamos National Laboratory Los Alamos, NM

Reported by Marsaret Watters, DECUS Scribe Service

John Crowell discussed the problems and the advantages of DIGITAL's debudging system, MACDBG. This program is a remote symbolic debussing tool which runs under RT-11SJ or RT-11XM on a FDP-11 or an LSI-11. This system has several features including the following: it loads programs into the target program via a Serial Line Unit (SLU); it examines and changes the target memory and registers; it has a RUN/HALT program; it is programmed to find breakpoints, watchpoints, and tracepoints; the Host console can be used as a target terminal; and it requires an ODT in the target. Finding the tracepoints is an especially useful tool in debussing, as the Potentially problematic point in the tarset program is indicated, yet the program continues to run, so the user can observe the effect the point has on the program.

This system also has some non-essential features that are helpful. There is a "Help" page; a Status display (on the VT100 only); a command key pad (on the VT100 and the LA120 only); programmable keys (VT100 and LA120 only); indirect command files; and a logging console I/O to file.

The Debus Service Module (DSM) is optional, however it is required for finding breakpoints, watchpoints, tracepoints, and for single stepping. The DSM resides in the Target RAM, and takes up 464 octal bytes. It contains a loader for moving blocks of data into target memory. There is also a handler for handling breakpoints, watchpoints, tracepoints, and for single stepping. The module speeds up loading and depositing in the target memory. This module should not be used if the user does not have the necessary RAM, or if he does not have RAM at locations 14 or 16. The program also should not be used if uses BPTs or if it used instructions that alter the T-bit. The DSM may be linked with the user's programs, and it can be loaded separately. The speaker warned that a user must be careful that

his program does not write over the DSM. He also warned that under SJ, the 'halt' instruction corrupts DSM (or at least MACDBG thinks it does).

Crowell pointed out several problems that he has noticed while using MACDBG. The major problem is that MACDBG sets the cursor keys to the "Application Code", which does not matter while MACDBG is in use, since it does not use the cursor keys. However, MACDBG does not reset the keys to "Cursor Code" before exiting. This particularly baffled Crowell. MACDBG has several buds of its own, but Crowell said that it is a powerful tool nonetheless, and that it is the most cost effective software that he has bought in a very long time.

Creation and Handling of Multi-Volume Directories Under RT-11 With TECO

#### Maarten van Swaay Kansas State University, Department of Computer Science Manhattan, KS

Jack Crowell, Session Chairperson Los Alamos National Laboratory Los Alamos, NM

Reported by J. Rick Mihalevich, DECUS Scribe Service

Many small RT-11 systems are based on floppy disk storage. Files and their backup copies can easily extend over 50-100 disk volumes, and locating an individual file can become a tedious chore of browsing through a large collection of disks or printed directories.

TECO can retrieve volume ID records and file names from a volume without invoking directory operations from USR. This carability makes it possible to use TECO for the creation of a multi-volume directory file. Because retrieval of the directory information does not invoke USR, the output volume can share a spindle with the input volumes from which the directories must be obtained.

A set of TECO macros for creation and use of a multi-volume directory file was described. The package presented includes provisions for the creation of a new directory file, for insertion / replacement / deletion of a single-volume directory in the file, and for locating selected files from the directory. Because the directory file can extend over more than half the space on a single volume, a mechanism was discussed to edit a large file-in-place. The presenter offered copies of these macros. To obtain a copy one needs to send a floppy to: Maarten Van Swaay, Kansas State University, Manhattan, Kansas 66506. The presenter requested that a package complete with return address and enough postage for return be included with the floppy.

# SOFTWARE PERFORMANCE REPORTS

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR	DOCUMENT TITLE	WEREION OR DOCUME	NTRADINO	DATE	
		araren Presentan en	DOLOMENT THEE	VERSION OR DOCUMENT PART NO.		DATE	
LBT-11	5.0	SIPP/PIP/B		V05.00		29-AUG	
			DEC OFFICE	AND CONTACT PERSON	DO YOU H	AVE SOURCE	
HAME: Ned W.			Lanh		YES	NO	
FIRM: E-System	ms, Melpar	Division	REPORT	REPORT TYPE/PRIORITY			
-	•		1X-	W/ERROR 2.	MODERATE 3	STEM IMP	
ADDRESS: 7700 Ar	lington Bly	vd.	SUGGES	TED ENHANCEMENT 3.	MINOR SYSTE	MIMPACT	
T-11- 0	hurch, Va.		OTHER	4.	X NO SIGNIFIC	NT IMPAC	
CUST. NO .: FAIIS C	nurch, va.	22040		5.	DOCUMENTAT	ION/SUGGES	
SUBMITTED BY:	PHO	NE:					
Gary L. Fuller		0-5000 X2858	CAN THE PR	OBLEM BE REPRODUCED	AT WILLT YES	X NO	
MAG TAPE FLOPPY DIS	TACHMENTS	DECTAPE	BETTER OR	SPR HAVE BEEN PREVEN MORE DOCUMENTATION	YES		
OTHER:			1				
CPU TYPE SERIAL	NO. MEMOR	Y SIZE DISTRIBUT	ION MEDIUM	SYSTEM DEVICE	DO NOT PUBL		
LSI-11/23 AB022	54 128 H	K RX-02		RL-01			
					and advertising the second		

1. A problem with the SIPP utility occurs when both of the following conditions are met:

- (1) An optional com-filespec is supplied in the SIPP command string.
- (2) A modification is made to the input file in the address range-1000 < ( Base + Offset ) ≤ 2000 (octal).</p>

The problem is characterized by the insertion of the command file text (destined for the com-filespec channel) into the input file starting at address 1000 (i.e. Block 1 ).

The probable diagnosis is that there is a channel mix-up when buffering the command file text. Why the 1000 - 2000 address range is a factor is undetermined.

The problem may be reproduced by performing any of the customization patches supplied in the RT-11 Installation Guide (AA-H376B-TC) Ch. 2.7 which specify addresses in the indicated range and by additionally specifying a com-filespec.

2. The /H and /V switches are transposed in the minireference. They are correct in the system utilities manual.

3. BACKUP/MULTI - If a file is too large to fit on the output volume and it is the last file being transferred, it continues to prompt for output volumes instead of giving a message that the file is too large.

## SYMPOSIUM TAPE INFORMATION

The second of the second se

### CALL FOR RT-11 SIG TAPE

### SUBMISSIONS

Assembling the RT-11 SIG Tapes at the DECUS Symposia (and producing a quality product) has turned out to be difficult. The Spring, 1983, tape was done after the Symposium, and I propose to do the same this time. Therefore, any SIG Tape submissions which are ready now can be sent to me for preparation. Please note, that even if you send a tape submission early, the DECUS U.S. Symposium Tape Copy Release Form <u>MUST BE SIGNED</u>!! A copy of the Release Form is attached below.

Please send all submittals, along with the Release Form to:

R. W. Barnard Sandia National Laboratories Division 2565A P. O. Box 5800 Albuquerque, NM 87185

Remember that the RT-11 SIG accepts not only 9-track, 800 bpi, magnetic tapes, but also RX01 and RX02 floppies. (I can also read TU-58 DECtape II's). Thank you.

ATTACHMENT D



Release Form Number:

#### TAPE COPY RELEASE AGREEMENT

The DECUS Program Library and the DECUS Tape Copy Facility provide a clearing house function only; programs are not sold or generated or tested. All programs and information and copies are provided "AS IS". DIGITAL EQUIPMENT COMPUTER USERS SOCIETY, DIGITAL EQUIPMENT CORPORATION, AND THE CONTRIBUTOR DISCLAIM ALL WARRANTIES ON THE PROGRAMS, INCLUDING WITHOUT LIMITATION, ALL IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS.

The following authorization is assumed for all programs copied on the copy facility:

Full permission and consent is hereby given to DECUS and to the DECUS Special Interest Group to reproduce, distribute, and publish and permit others to reproduce in whole or in part, in any form and without restriction, this program and any information relating thereto. The submitter hereby warrants and represents that he had good and sufficient right, interest, and title in and to this program and the related information to grant such permission to DECUS.

Date

ATTACHMENT C



Release Form Number:

## DECUS U.S. SYMPOSIUM TAPE COPY RELEASE FORM

Name				
Company				
Address				
State	State		phone	
Program Name(s)				
	□ VAX □ STRUCT. LANC	□RT-11 G.	□ TOPS-10 □ TOPS-20	
Contents of Tape:				
Number of Files		PPN		
Is this material account spec				
Number and Kinds of Tape Submitted:	<ul> <li>DOS Format</li> <li>7-track</li> <li>800 BPI</li> </ul>	<ul> <li>ANSI Format</li> <li>9-track</li> <li>1600 BPI</li> </ul>	Other     Other     Other     Other	
Description:				

### Guidelines:

Users who wish to participate in the exchange should bring a 2400 foot (preferably new) quality tape to the Symposium. The tape and cannister should be clearly labeled with the user's name and address.

- 1. No proprietary or licensed software (including whole or partial copies) may be submitted.
- 2. Users who would like to submit modifications to licensed DIGITAL software may submit files to be appended to the original source program. ONLY the modifications may be submitted.
- 3. Users are encouraged to include a README file on their tape including the submitter's name and address, and a description of the files he/she is submitting.
- 4. Tapes should be compatible with standard system software. Please indicate the number of files and the PPN, UIC or account, and tape format.
- 5. Tapes should be 9-track, and be labeled with the sender's name and address.

IMPORTANT! RELEASE AGREEMENT ATTACHED